UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA
CORSO DI LAUREA MAGISTRALE IN MATEMATICA

TESI DI LAUREA

SULLA CRITTOGRAFIA OMOMORFA

RELATORI: PROF. A. LANGUASCO, PROF. A. TONOLO

DIPARTIMENTO DI MATEMATICA

LAUREANDA: STEFANIA LIPPIELLO

MATRICOLA: 621556

ANNO ACCADEMICO 2012/2013

Dood qrqqd h d Oxfd

Se pareba boves,
alba pratàlia aràba,
et albo versòrio teneba,
et negro sèmen seminaba.

Indovinello veronese
(VIII-IX sec.)

Introduzione

Immaginiamo la seguente situazione: Alice manda a Bob una valigetta chiusa con un lucchetto e gli chiede di contare il denaro contenuto all'interno.
"Sicuro", risponde Bob, "dammi la chiave", ma Alice scuote la testa, non fidandosi ciecamente di lui. Bob allora scrolla la valigetta cercando di giudicarne il peso, prova ad avvicinare l'orecchio, ma alla fine conclude affermando "non si può fare senza poter accedere all'interno".
Questo esempio, nella sua estrema semplicità, introduce il problema su cui si fonda la nostra intera discussione: si possono effettuare operazioni su dati che non si possono vedere? O meglio su dati codificati senza alterare il risultato?
Facciamo un ulteriore esempio per chiarire meglio la questione. Supponiamo che Alice dia a Bob un file cifrato che contiene una lista di numeri e gli chieda di sommarli, ma senza fornirgli la chiave di decifratura. Di nuovo Bob si trova in difficoltà: il file cifrato ha preso il posto della valigetta e anche questa volta Bob non sa rispondere, non potendo conoscere il messaggio "in chiaro" (ossia la lista di numeri).
Ma Alice ha scelto un particolare crittosistema che permette a Bob di lavorare ugualmente sui dati codificati e fornire il risultato cifrato delle operazioni richieste da Alice. Utilizzando poi la chiave di decifratura, Alice otterrà la somma desiderata.
Ciò è possibile perché il crittosistema usato da Alice è pienamente omomorfo. Possiamo così introdurre il concetto di *fully homomorphic encryption scheme* (FHE), dove "fully" sta ad indicare che non ci sono limitazioni a quali manipolazioni possono essere svolte sui dati codificati.

Contestualizziamo il problema suddetto in un quadro più generale.
Innanzitutto un'importante rivoluzione nella disciplina della Crittografia si ebbe nel 1976 quando Diffie ed Hellman introdussero il concetto di Crittografia a chiave pubblica, detta anche asimmetrica per mettere in risalto il diverso ruolo che hanno la chiave di cifratura, che viene resa pubblica, e la chiave di decifratura, che invece rimane segreta. Per la prima volta nella

storia si è avuto un crittosistema che non fonda la propria sicurezza sull'assoluta segretezza delle tecniche di codifica ed anzi le rende pubbliche.
Citiamo uno dei più popolari crittosistemi a chiave pubblica, RSA, dalle iniziali di Rivest, Shamir e Adleman che lo proposero nel 1978 in [1] e descriviamolo brevemente.
Alice, per codificare il suo messaggio, deve svolgere prima di tutto le seguenti operazioni:

1. sceglie due numeri primi p e q, distinti e sufficientemente grandi;
2. calcola $n = p \cdot q$;
3. calcola $\varphi(n) = (p-1)(q-1) = n - p - q + 1$;
4. sceglie $e \in \mathbb{N}$ tale che $(e, \varphi(n)) = 1$;
5. determina $d \in \mathbb{Z}^*_{\varphi(n)}$ tale che $e \cdot d \equiv 1 \bmod \varphi(n)$;
6. rende nota la coppia (n, e), che è la sua chiave pubblica;
7. tiene segreti p, q e d, che costituiscono la sua chiave segreta.

La funzione crittografica di Alice è $\mathsf{Enc}_{n,e}(x) = x^e \bmod n$, che può essere calcolata da tutti gli utenti del sistema, mentre la funzione che utilizza per decifrare è $\mathsf{Dec}_d(y) = y^d \bmod n$, per cui è necessario conoscere d e quindi $\varphi(n)$, ovvero la fattorizzazione di n.

In tale contesto si colloca l'intuizione di avere un crittosistema pienamente omomorfo. L'idea risale al 1978; ne troviamo menzione infatti nell'articolo [26] pubblicato da Ronald L. Rivest, Len Adleman e Michael L. Dertouzos, pochi mesi dopo la prima implementazione di RSA. Essi suggerirono la possibilità di creare un crittosistema pienamente omomorfo con l'obiettivo di aumentare la privacy sui dati, senza tuttavia riuscire a fornirne un esempio sicuro. Tale risultato è stato raggiunto solo recentemente, nel 2009, grazie al lavoro di Craig Gentry, laureato alla Stanford University e ora ricercatore per l'IBM. In breve l'idea è la seguente: dati i testi cifrati $c_1, \ldots, c_t$ corrispondenti ai messaggi in chiaro $m_1, \ldots, m_t$, operazioni quali addizioni e moltiplicazioni su $c_1, \ldots, c_t$ forniscono un output $c = f(c_1, \ldots, c_t)$ che corrisponde al testo cifrato di $f(m_1, \ldots, m_t)$ ottenuto svolgendo le medesime operazioni su $m_1, \ldots, m_t$.
Abbiamo riportato la descrizione di RSA non solo per completezza, ma anche perché costituisce un esempio di crittosistema parzialmente omomorfo, nel senso che permette di svolgere

il prodotto dei testi cifrati mantenendo la corrispondenza con il testo in chiaro, ma non l'addizione:
siano $c_1 = \mathsf{Enc}_{n,e}(m_1) = m_1^e \bmod n$ e $c_2 = \mathsf{Enc}_{n,e}(m_2) = m_2^e \bmod n$, allora

$$c_1 \cdot c_2 = (m_1^e \cdot m_2^e) \bmod n = (m_1 \cdot m_2)^e \bmod n = \mathsf{Enc}_{n,e}(m_1 \cdot m_2).$$

È immediato notare che RSA non è omomorfo rispetto alla somma.
A titolo esemplificativo diamo un'ulteriore analogia con il mondo fisico che lo stesso Gentry riporta in [13]. Immaginiamo che Alice possieda una gioielleria e debba assemblare dei materiali preziosi come oro, argento e diamanti, per ottenere degli intricati anelli e complesse collane. Tuttavia ella non si fida dei suoi dipendenti e vorrebbe che essi fossero in grado di creare i gioielli senza avere accesso diretto ai materiali. Per questo motivo costruisce una scatola con dei guanti, chiusa con un lucchetto di cui solo Alice possiede la chiave, e al cui interno pone i materiali preziosi. I lavoratori attraverso i guanti possono assemblare anelli e collane, ma non possono portar via l'oro o le pietre preziose.
Nell'esempio la scatola impenetrabile rappresenta la cifratura dei messaggi in chiaro a cui si può avere accesso solo con la chiave segreta, mentre i guanti svolgono il ruolo dell'omomorfismo (o malleabilità) del crittosistema. I gioielli ottenuti sono inaccessibili finché sono dentro la scatola, solo Alice, che possiede la chiave del lucchetto, potrà utilizzarli; essi incarnano pertanto la cifratura della funzione svolta sui dati iniziali, $f(m_1, \ldots, m_t)$. Osserviamo che l'impossibilità di acquisizione dei dati è ottenuta tramite la privazione dell'accesso fisico ai gioielli, in opposizione all'accesso visivo dei messaggi in chiaro.
Naturalmente la gioielleria di Alice è solo un esempio, non riesce a rendere tutti gli aspetti della crittografia omomorfa e perciò non va presa alla lettera, ma è utile a darci un'idea del problema e della sua soluzione; pertanto in seguito la riutilizzeremo per chiarire ulteriori concetti.

Diamo qualche dettaglio maggiormente tecnico. Un crittosistema $\mathcal{E}$ è dotato di tre algoritmi: $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$ e $\mathsf{Decrypt}_{\mathcal{E}}$. Nei crittosistemi simmetrici $\mathsf{KeyGen}_{\mathcal{E}}$ genera una singola chiave che viene utilizzata sia in $\mathsf{Encrypt}_{\mathcal{E}}$ che in $\mathsf{Decrypt}_{\mathcal{E}}$, mentre nei crittosistemi asimmetrici $\mathsf{KeyGen}_{\mathcal{E}}$ genera due chiavi, una chiave di cifratura pubblica, pk, e una chiave di decifratura segreta, sk. Un crittosistema omomorfo può essere sia simmetrico che asimmetrico, ma noi ci concentreremo sul secondo caso; inoltre tratteremo un tipo di crittografia detto probabilistico, introdotto nel 1982 da Shafi Goldwasser e Silvio Micali. A differenza dei sistemi deterministici, per i quali messaggi in chiaro uguali producono lo stesso testo cifrato, nei crittosistemi probabilistici la codifica dello stesso messaggio svolta in momenti diversi produce testi cifrati

differenti; questo è possibile grazie all'inserimento di un fattore aleatorio, che chiameremo "rumore", all'interno della fase di codifica. Schematizzando, dati un messaggio in chiaro m e una funzione di cifratura Enc, siano $\mathsf{Enc}(m) = c_1$ la prima codifica di m e $\mathsf{Enc}(m) = c_2$ la seconda codifica di m, risulta in genere $c_1 \neq c_2$.

La peculiarità della crittografia omomorfa sta nella presenza di un quarto algoritmo, $\mathsf{Evaluate}_{\mathcal{E}}$, associato ad un insieme $\mathcal{F}_{\mathcal{E}}$ di funzioni permesse. Per ogni funzione f in $\mathcal{F}_{\mathcal{E}}$ e per ogni testo cifrato $c_1, \ldots, c_t$ con $c_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(m_i)$, l'algoritmo $\mathsf{Evaluate}_{\mathcal{E}}(f, c_1, \ldots, c_t)$ fornisce come output un testo cifrato c tale che $\mathsf{Decrypt}_{\mathcal{E}}(c) = f(m_1, \ldots, m_t)$.

Il crittosistema $\mathcal{E}$ è pienamente omomorfo se, a prescindere dalla complessità della funzione $f \in \mathcal{F}_{\mathcal{E}}$, non viene inficiata la correttezza della decodifica e non varia il tempo di decifratura dell'output finale di $\mathsf{Evaluate}_{\mathcal{E}}$. Utilizzando l'esempio della gioielleria, i dipendenti di Alice possono assemblare complicatissimi gioielli, ma questo non influisce sul tempo necessario ad Alice per estrarli dalla scatola.

Ma come possiamo stimare la complessità di f? La risposta più ovvia è utilizzare il tempo di esecuzione T_f di una macchina di Turing che computi f. Utilizziamo pertanto una misura ad esso correlato, la dimensione S_f di un circuito booleano (i.e. il numero di porte AND, OR e NOT) che calcoli f.

Richiedere che il crittosistema sia allo stesso tempo probabilistico e pienamente omomorfo genera un conflitto: i testi cifrati con un processo di codifica probabilistica sono contaminati da un rumore, che li rende meno attaccabili, ma che viene amplificato da ogni operazione svolta sui testi. Se la funzione $f \in \mathcal{F}_{\mathcal{E}}$ è troppo complessa, il rumore può inquinare troppo il risultato rendendolo in tal modo inutilizzabile. Pensiamo ad ogni valore del testo cifrato come ad un punto nello spazio. La funzione di codifica probabilistica inietta un po' di casualità nelle coordinate dei punti spostandoli leggermente dalla posizione che occuperebbero in un sistema deterministico. La funzione di decifratura prima filtra il rumore trattando ogni punto come se fosse locato nella posizione imperturbata più vicina e poi procede alla decodifica vera e propria (si veda la Figura 0.1).

Quando però il rumore viene amplificato dalle operazioni effettuate tanto da spostare troppo il punto, la filtrazione del rumore lo associa ad un punto errato, non essendo più in grado di risalire alla sua posizione corretta. Quindi è necessario limitare il numero di operazioni al fine di arginare l'accumulo di errore; in questo modo però non possiamo più parlare di crittosistema *pienamente omomorfo*, ma solo *"quasi"* o *"parzialmente" omomorfo* (somewhat homomorphic).

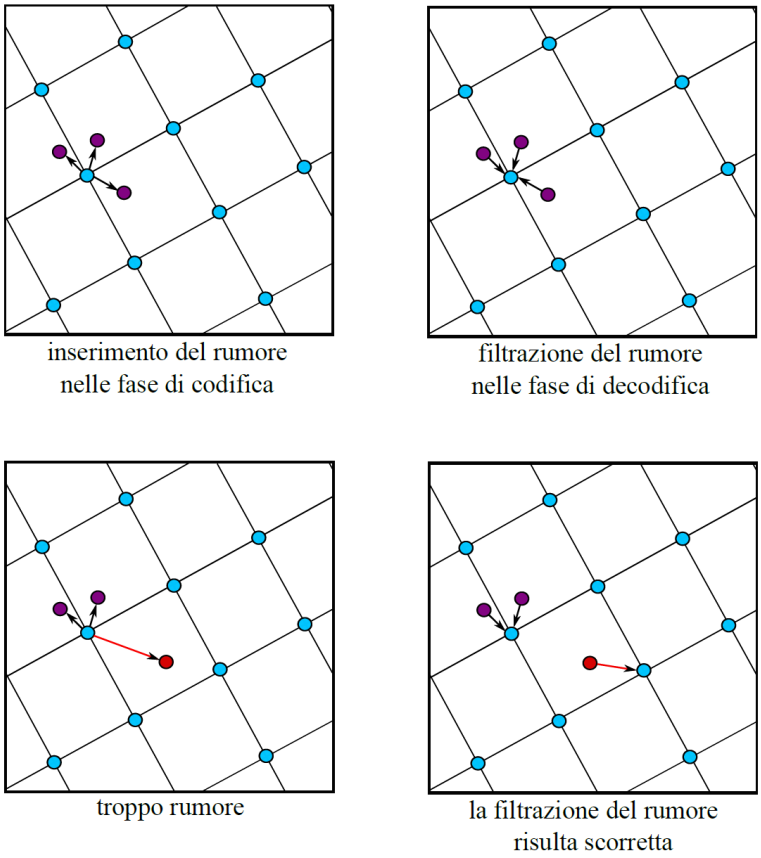

Figura 0.1: I dati cifrati posso essere visualizzati come punti (*cerchi viola*) che si trovano leggermente spostati rispetto ai punti di un fissato reticolo (*cerchi azzurri*).

Vedremo tuttavia come è possibile passare da un crittosistema parzialmente omomorfo ad uno che lo sia pienamente: è proprio questa la grande innovazione fatta da Gentry.
Continuando ad usare l'analogia col mondo fisico di Alice e della sua gioielleria, il problema diventa il seguente. Alice riceve delle scatole difettose, dopo un minuto di lavoro i guanti diventano rigidi e così inutilizzabili. Ma Alice trova una soluzione: come prima dà una scatola chiusa con i materiali all'interno (scatola1) ad un dipendente, ma gli dà anche un'altra scatola (scatola2) contenente la chiave della scatola1 e una scatola3 contenente la chiave della scatola2 e così via. Per assemblare un gioiello, il lavoratore utilizzerà la scatola1 finché i guanti non si irrigidiranno, dopodiché metterà la scatola1 nella scatola2 dove trova la chiave per aprirla e potrà così continuare l'assemblaggio fino a quando anche i guanti della scatola2 si romperanno e allora passerà alla scatola3 e così via. Nell'esempio le scatole difettose rappresentano il nostro crittosistema quasi omomorfo, che può svolgere le operazioni di somma e moltiplicazione sui testi cifrati solo finché il rumore non diventa troppo grande. Quello che vorremmo fare è usare

questo crittosistema per costruirne uno pienamente omomorfo.
Uscendo dalle metafore, per poter ripulire i dati dal rumore che hanno acquisito dopo un certo numero di operazioni, il nostro algoritmo $\mathsf{Evaluate}_{\mathcal{E}}$ deve essere in grado di maneggiare anche la funzione di decifratura, oltre a svolgere le operazioni di somma e moltiplicazione.
Nell'analogia Alice ha scoperto che c'è un'unica cosa che i suoi dipendenti devono essere in grado di fare in meno di un minuto oltre a continuare l'assemblaggio del gioiello: sbloccare una scatola con la successiva ed estrarre il pezzo.
Se $\mathcal{E}$ possiede la proprietà di saper utilizzare la propria funzione di decifratura (o meglio una sua versione leggermente incrementata), diciamo che il crittosistema $\mathcal{E}$ è *bootstrappable*. Se $\mathcal{E}$ è bootstrappable allora possiamo passare da un crittosistema quasi omomorfo ad uno pienamente omomorfo.
Purtroppo la soluzione suggerita da Alice, pur funzionando, rallenta di molto la produzione di gioielli. Questo è infatti il problema ancora da risolvere: il crittosistema proposto da Gentry funziona ed è sicuro, ma richiede tempi di esecuzione molto lunghi o testi cifrati estremamente più grandi dei relativi testi in chiaro.
Ci sono ancora delle sfide da superare affinché un crittosistema pienamente omomorfo possa essere utilizzato in pratica, ma numerosi passi sono stati fatti in questa direzione negli ultimi anni e, nel caso si riuscisse a migliorare l'efficienza di tali metodi, la strada che si aprirebbe sarebbe ricca di interessanti applicazioni.
Gli ambiti in cui impiegare tale risultato sono diversi, dalla medicina alla finanza, dal marketing a una semplice ricerca su dati personali. Immaginiamo ad esempio uno scenario futuro in cui un uomo possa monitorare i suoi parametri fisiologici e, fornendoli cifrati ad un server, possa ricevere analisi dei propri dati che solo lui sia in grado di vedere, tutelando la propria privacy.
È notevole come un'idea avuta negli anni Settanta stia risultando finalmente tangibile, a distanza di più di trent'anni; questo mette in luce come la lungimiranza di certe personalità porti sviluppi altrimenti inimmaginabili.

Outline Nel presente lavoro non seguiremo l'evoluzione cronologica dei crittosistemi omomorfi, partendo dal primo risultato ottenuto da Gentry nel 2009; inizieremo, invece, col descriverne uno più recente basato sull'aritmetica modulare. Rispetto al suo predecessore, che si basa sulla geometria dei reticoli, risulta concettualmente più semplice; in tal modo potremo concentrarci maggiormente sulla comprensione del metodo utilizzato per passare da un crittosistema parzial-

mente omomorfo ad uno pienamente omomorfo, che è lo stesso seguito da Gentry in [12]. Lo schema che tratteremo nel Capitolo 1 si fonda sul lavoro [10] svolto da M. van Dijk, C. Gentry, S. Halevi e V. Vaikuntanathan nel 2010. Grazie all'analisi di tale crittosistema, capiremo che la grande innovazione di Gentry si basa sulla suddivisione della costruzione dello schema in tre passi:

1. si costruisce un crittosistema parzialmente omomorfo, ovvero capace di valutare solo un numero limitato di operazioni sui dati codificati, corrispondenti a circuiti di profondità limitata;

2. si comprime il circuito di decodifica del crittosistema ottenuto al punto 1, in modo che il crittosistema sia in grado di valutarne una sua versione leggermente aumentata (il crittosistema è diventato bootstrappable);

3. si fornisce una procedura di "refresh" per pulire i dati dagli errori accumulati dalle varie operazioni, che causerebbero una decodifica scorretta, e si ottiene un crittosistema pienamente omomorfo.

Prima di descrivere il crittosistema basato sui reticoli, avremo bisogno di costruire le fondamenta sulle quali si regge. Dedicheremo il Capitolo 2 alla definizione dei reticoli, alla dimostrazione dei risultati necessari e a delineare i problemi computazionali utilizzati per garantire la sicurezza dei sistemi crittografici analizzati. Nel Capitolo 3 tratteremo il crittosistema scritto da Gentry in [12] nel 2009 e nel Capitolo 4 riporteremo la sua implementazione a seguito di una serie di ottimizzazioni dovute sempre a Gentry insieme a Halevi [15]. Purtroppo i risultati mettono in luce come si sia ancora lontani da rendere la crittografia omomorfa sufficientemente efficiente da poter essere utilizzata. Nel Capitolo 5 parleremo di quali strade sono state attualmente intraprese perché ciò si realizzi, introducendo il crittosistema BGV [5], il Chimeric FHE [16] e una libreria open source che implementa la crittografia omomorfa (HElib).

Indice

Capitolo 1

Fully homomorphic encryption over the integers

1.1 Definizioni

Un crittosistema asimmetrico ordinario $\mathcal{E}$ consiste di tre algoritmi: $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$ e $\mathsf{Decrypt}_{\mathcal{E}}$. $\mathsf{KeyGen}_{\mathcal{E}}$ è un algoritmo randomizzato che prende in input un parametro di sicurezza λ e fornisce come output una chiave segreta sk e una chiave pubblica pk. $\mathsf{Encrypt}_{\mathcal{E}}$ è un algoritmo randomizzato che ha come input pk e un messaggio in chiaro m e da esso risulta il testo cifrato c. $\mathsf{Decrypt}_{\mathcal{E}}$, infine, dati sk e c, ci restituisce il messaggio in chiaro m.
La complessità computazionale di questi algoritmi deve essere polinomiale in λ.
La correttezza è definita come segue: dati $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\mathcal{R}} \mathsf{KeyGen}_{\mathcal{E}}$, un messaggio in chiaro m e $c \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}$ allora $\mathsf{Decrypt}_{\mathcal{E}} \to m$.
Oltre ai suddetti algoritmi, un crittosistema omomorfo $\mathcal{E}$ ne possiede un altro (possibilmente randomizzato), $\mathsf{Evaluate}_{\mathcal{E}}$, che fornisce un ulteriore testo cifrato c, dati la chiave pubblica pk, un circuito C preso da un insieme di circuiti permessi $\mathcal{C}_{\mathcal{E}}$ e una t-upla di testi cifrati $\langle c_0, \ldots, c_t \rangle$, che sono input del circuito C.
Specifichiamo che con circuito intendiamo un grafo aciclico diretto in cui distinguiamo i nodi di ingresso o di input (privi di archi entranti e etichettati con una variabile booleana), i nodi operazione o porte (dotati di archi entranti e uscenti, scelti tra le porte AND, OR o NOT) e i nodi di uscita o di output (privi di archi uscenti).
I nodi sono collegati tra loro da archi orientati, ma non sono presenti circoli chiusi, cioè percorsi composti da archi che partono e terminano nello stesso nodo. Su di esso è possibile creare un ordinamento delle porte: se un nodo A è unito ad un altro B con un arco orientato che parte da A e arriva a B, si dice che A precede B, in tal modo possiamo numerare i nodi, a partire

dallo 0 che contrassegna i nodi di input; inoltre l'arco orientato prende il numero del nodo in cui entra. Così facendo si ottiene una partizione del circuito in livelli.
La dimensione di un circuito indica il numero di porte in esso contenute e la sua profondità consiste nella lunghezza, espressa in numero di archi, del percorso più lungo tra input e output.
D'ora in poi useremo solo circuiti qualificati come grafi aciclici diretti e pertanto ci riferiremo agli archi orientati chiamandoli soltanto archi.
Definiamo inoltre fan-in come il numero di archi entranti in un nodo e fan-out il numero di archi uscenti da un nodo.

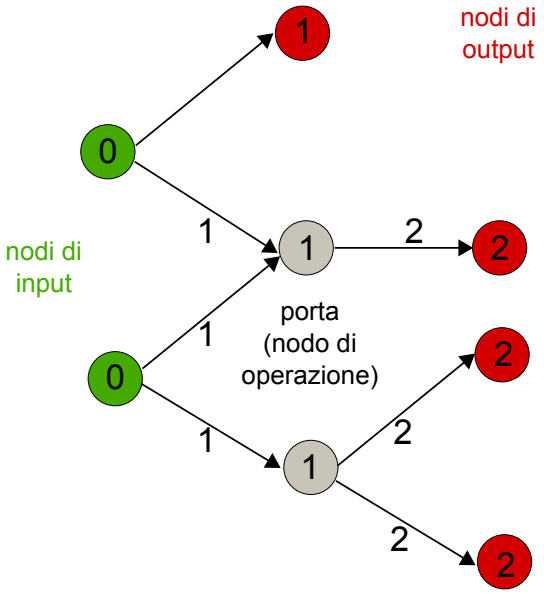

Figura 1.1: Il circuito disegnato è un grafo aciclico diretto di dimensione 8 e profondità 2.

Definizione 1.1. *Diciamo che un crittosistema omomorfo $\mathcal{E}$ è* ***corretto*** *per i circuiti in $\mathcal{C}_{\mathcal{E}}$ se per ogni coppia di chiavi $(\mathsf{sk}, \mathsf{pk})$ fornita da $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, per ogni circuito $C \in \mathcal{C}_{\mathcal{E}}$, per ogni testo in chiaro $m_1, \ldots, m_t$ e ogni testo cifrato $c_1, \ldots, c_t$ con $c_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}, m_i)$ si ha:*

$$\text{se } c \leftarrow \mathsf{Evaluate}_{\mathcal{E}}(\mathsf{pk}, C, \langle c_1, \ldots, c_t \rangle) \text{ allora } \mathsf{Decrypt}_{\mathcal{E}}(\mathsf{sk}, c) \rightarrow C(m_1, \ldots, m_t)$$

eccetto che per una probabilità trascurabile sulle eventuali scelte casuali effettuate in $\mathsf{Evaluate}_{\mathcal{E}}$.

Da sola la mera correttezza non esclude i casi banali; in particolare supponiamo di definire $\mathsf{Evaluate}_{\mathcal{E}}(\mathsf{pk}, C, \langle c_1, \ldots, c_t \rangle)$ in modo che dia come output semplicemente $(C, \langle c_1, \ldots, c_t \rangle)$ senza processare i testi cifrati tramite il circuito, poi $\mathsf{Decrypt}_{\mathcal{E}}$ decifrerà i testi cifrati e applicherà C al risultato. Questo schema è corretto, ma non è interessante. Per rimediare a questa mancanza

diamo un limite superiore alla lunghezza dei testi cifrati risultanti da $\mathsf{Evaluate}_{\mathcal{E}}$. Un modo per attuare ciò è dare un limite superiore alla dimensione del circuito di decodifica $D_{\mathcal{E}}$ per il crittosistema $\mathcal{E}$; tale limite dipende solo dal parametro di sicurezza, come nella definizione seguente.

Definizione 1.2. *Diciamo che un crittosistema omomorfo $\mathcal{E}$ è **compatto** se esiste un polinomio f tale che, per ogni valore del parametro di sicurezza λ, l'algoritmo di decodifica di $\mathcal{E}$ possa essere espresso da un circuito $D_{\mathcal{E}}$ di dimensione al massimo $f(\lambda)$.*

Definizione 1.3. *Un crittosistema omomorfo $\mathcal{E}$ **valuta in modo compatto** i circuiti in $\mathcal{C}_{\mathcal{E}}$ se $\mathcal{E}$ è compatto e corretto per i circuiti in $\mathcal{C}_{\mathcal{E}}$.*

Possiamo ora definire precisamente un crittosistema pienamente omomorfo.

Definizione 1.4. *Un crittosistema omomorfo $\mathcal{E}$ si dice **pienamente omomorfo** (o **fully homomorphic**) se valuta in modo compatto tutti i possibili circuiti.*

Definizione 1.5. *Consideriamo un crittosistema omomorfo $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Evaluate}, \mathsf{Decrypt})$, in cui la decodifica sia implementata da un circuito che dipenda solo dal parametro di sicurezza (questo significa che, per un fissato valore del parametro di sicurezza, la dimensione della chiave segreta è sempre la stessa e, analogamente, anche tutti i testi cifrati). Per un dato valore del parametro di sicurezza λ, l'**insieme dei circuiti di decifratura aumentati** consiste in due circuiti, entrambi presi come input, una chiave segreta e due testi cifrati: un circuito decodifica entrambi i testi cifrati e somma modulo 2 i risultanti bit in chiaro, l'altro decifra i testi e moltiplica i bit risultanti modulo 2. Denotiamo questo insieme con $D_{\mathcal{E}}(\lambda)$.*

Definizione 1.6. *Sia $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Evaluate}, \mathsf{Decrypt})$ un crittosistema omomorfo e, per ogni valore del parametro di sicurezza λ, sia $\mathcal{C}_{\mathcal{E}}(\lambda)$ un insieme di circuiti rispetto al quale $\mathcal{E}$ è corretto. Diciamo che $\mathcal{E}$ è **bootstrappable** se $D_{\mathcal{E}}(\lambda) \subseteq \mathcal{C}_{\mathcal{E}}(\lambda)$ vale per ogni λ.*

1.2 A somewhat homomorphic encryption scheme

Consideriamo ora un crittosistema che sia omomorfo rispetto a circuiti booleani che consistono delle porte per l'addizione e la moltiplicazione modulo 2 e osserviamo che effettuare solo operazioni sui bit comporta che lo spazio dei messaggi in chiaro scelto è limitato a $\{0, 1\}$.

1.2.1 Parametri

Precisiamo, innanzitutto, che utilizzeremo le lettere dell'alfabeto greco per indicare i parametri; in particolare, abbiamo già incontrato λ, che individua il parametro di sicurezza. Tutti i logaritmi, se non diversamente specificato, sono intesi in base 2 e, inoltre, dato un numero reale z denotiamo con $\lceil z \rceil$, $\lfloor z \rfloor$ e $\lfloor z \rceil$ rispettivamente la parte intera superiore (funzione ceiling), la parte intera (funzione floor) e l'intero più vicino; essi sono, nell'ordine, gli interi in $[z, z+1)$, $(z-1, z]$ e $(z-1/2, z+1/2]$.
Infine, per un numero reale z e un intero p, siano $q_p(p) = \lfloor z/p \rceil$ e $r_p(z) = z - q_p(z)$; in particolare consideriamo $z \bmod p = r_p(z)$ e tale valore risulta nell'intervallo $(-p/2, p/2]$. La costruzione che segue ha diversi parametri che controllano il numero degli interi nella chiave pubblica e il numero di bit dei vari interi; nello specifico useremo i seguenti (tutti polinomiali nel parametro di sicurezza λ):

- γ è il numero di bit degli interi nella chiave pubblica;
- η è il numero di bit della chiave segreta;
- ρ è il numero di bit del rumore (la distanza tra gli elementi della chiave pubblica e i multipli più vicini della chiave segreta);
- τ è il numero di interi della chiave pubblica.

Questi parametri devono soddisfare i seguenti vincoli il cui significato sarà chiarito nel seguito:

- $\gamma = \omega(\eta^2 \log \lambda)$;
- $\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$;
- $\rho = \omega(\log \lambda)$;
- $\tau \geq \gamma + \omega(\log \lambda)$.

Usiamo inoltre un parametro secondario per il rumore: $\rho' = \rho + \omega(\log \lambda)$.

Osservazione 1.7. *Ricordiamo la definizione di ω e Θ utilizzate per i nostri parametri.*

- *Si dice che $f(n) = \omega(g(n))$ se $\lim_{n\to+\infty} \left|\frac{f(n)}{g(n)}\right| = \infty$, ossia $g(n) = o(f(n))$.*

- $f(n) = \Theta(g(n))$ *se* $0 < \liminf_{n\to+\infty} \left|\frac{f(n)}{g(n)}\right| \leq \limsup_{n\to+\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$ *oppure, equivalentemente, se* $\exists c_1, c_2 > 0$ *e* $n_0 \in \mathbb{N}$ *tali che* $\forall n > n_0$ *si ha che* $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$.

- *Si definisce* $f(n) = \widetilde{O}(g(n))$ *se* $f(n) = O(g(n)\log^k g(n))$ *per qualche* k, *in cui* $f(n) = O(g(n))$ *se* $\exists n_0, \exists C > 0$ *t.c.* $|f(n)| \leq C|g(n)|$ *per* $n > n_0$.

- $f(n) \asymp g(n)$ *se e solo se* $f(n) = \widetilde{O}(g(n))$ *e* $g(n) = \widetilde{O}(f(n))$.

Infine utilizzeremo $\ll$ *con l'accezione di "molto minore di".*

Parametri scelti Una conveniente scelta dei parametri che verrà da noi adottata è la seguente:

$$\begin{aligned} \rho &= \lambda, \\ \rho' &= 2\lambda, \\ \eta &\asymp \lambda^2, \\ \gamma &\asymp \lambda^5, \\ \tau &= \gamma + \lambda. \end{aligned} \tag{1.1}$$

Osservazione 1.8. *Dato un circuito C composto di porte per l'addizione e la moltiplicazione applicate ai bit, ossia mod 2, consideriamo la sua generalizzazione agli interi, ovvero lo stesso circuito C con l'addizione e la moltiplicazione applicate però agli interi piuttosto che ai bit. In generale, ogniqualvolta che estenderemo le operazioni svolte sugli input, chiameremo il circuito* ***generalizzato*** *e sarà indicato con $C^\dagger$ o $g(C)$.*

Possiamo ora definire con precisione un circuito permesso e il suo legame con il polinomio calcolato dal circuito stesso.

Definizione 1.9. *Definiamo un* ***circuito permesso*** *uno in cui, per ogni $\alpha \geq 1$ e ogni insieme di input interi tutti minori di $2^{\alpha(\rho'+2)}$ in valore assoluto, vale che l'output del circuito generalizzato ha valore assoluto al massimo di $2^{\alpha(\eta-4)}$. Indichiamo con $\mathcal{C}_\mathcal{E}$ l'insieme dei circuiti permessi.*

Dato un circuito booleano C con t input, indichiamo con $C^\dagger$ il suo circuito generalizzato (con le porte booleane sostituite da operazioni intere) e il polinomio $f(x_1, \ldots, x_t)$ in t variabili

calcolato da $C^\dagger$. Osserviamo che se $|\vec{f}| \cdot (2^{\rho'+2})^d \leq 2^{\eta-4}$, dove $|\vec{f}|$ è la norma l_1 del vettore formato dai coefficienti di f, allora $C \in \mathcal{C}_\mathcal{E}$. In particolare

$$d \leq \frac{\eta - 4 - \log|\vec{f}|}{\rho' + 2}. \tag{1.2}$$

Chiamiamo *polinomi permessi* i polinomi che soddisfano la relazione (1.2). Essi formano un insieme denotato con $\mathcal{P}_\mathcal{E}$ e indichiamo i circuiti che li computano $C(\mathcal{P}_\mathcal{E})$. Da quanto detto segue che $C(\mathcal{P}_\mathcal{E}) \subseteq \mathcal{C}_\mathcal{E}$.

Osservazione 1.10. *Per i nostri scopi, consideriamo* $\log|\vec{f}|$ *piccolo rispetto a* η, $\rho' = \omega(\log\lambda)$ *e* $t, \tau \leq \lambda^\beta$ *e abbiamo bisogno di supportare polinomi di grado fino a* $\alpha\lambda\log^2\lambda$ *per qualche costante positiva* α, β. *Mettendo in relazione quanto detto con l'espressione (1.2), è sufficiente porre* $\eta = \rho' \cdot \Theta(\lambda\log^2\lambda)$.

1.2.2 La costruzione

Possiamo illustrare ora un esempio di crittosistema quasi omomorfo.

KeyGen(λ): la chiave segreta è un intero dispari di η bit, $p \leftarrow (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^\eta)$.
Per la chiave pubblica, invece, definiamo prima di tutto una procedura per ottenere degli interi x a partire da p:

- choose $q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p)$,
- $r \xleftarrow{\mathcal{R}} \mathbb{Z} \cap (-2^\rho, 2^\rho)$,
- output: $x = pq + r$.

I risultati ottenuti dalla suddetta procedura formano una distribuzione, che indicheremo con $\mathcal{D}_{\gamma,\rho}(p)$. Da questa distribuzione si prelevano casualmente $x_i = q_i p + r_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$ per $i = 0, \ldots, \tau$. Rietichettate le x_i in modo che $|x_0|$ sia la più grande, si ricomincia fino a quando x_0 non sia dispari e $r_p(x_0)$ non sia pari. La chiave pubblica allora è $\mathsf{pk} = \langle x_0, \ldots, x_\tau \rangle$.

Encrypt($\mathsf{pk}, m \in \{0,1\}$): sceglie un sottoinsieme random $S \subseteq \{1, 2, \ldots, \tau\}$ e un intero random $r \in (-2^{\rho'}, 2^{\rho'})$ e fornisce come output $c \leftarrow (m + 2r + 2\sum_{i\in S} x_i) \bmod x_0$.

Evaluate$(\mathsf{pk}, C, c_1, \ldots, c_t)$: dato il circuito (binario) C con t input e t testi cifrati c_i, applica le porte di addizione e moltiplicazione (intera) di $C^\dagger$ ai testi cifrati, svolgendo tutte le operazioni sugli interi e fornendo il risultato intero.

Decrypt(sk, c): dà come output $m' \leftarrow (c \bmod p) \bmod 2$.

Esempio. Diamo un piccolo esempio numerico, banale, ma che possa aiutare a chiarire il crittosistema descritto.
Siano $p = 131$ e $x_0 = 131 \cdot 3 + 2 = 395$, $x_1 = 131 \cdot 1 + 1 = 132$, $x_2 = 131 \cdot 2 - 1 = 261$, $x_3 = 131 \cdot 1 - 3 = 128$. Consideriamo $m = 1$ e calcoliamo due sue possibili codifiche:

$$\begin{aligned} c_1 &= (1 + 2 \cdot 4 + 2 \cdot (132 + 128)) \bmod x_0 = 131 \cdot 4 + (2 \cdot 2 + 1) = 529 \bmod 395 = 134, \\ &\qquad\qquad (\text{rumore} = 5 < p/2) \\ c_2 &= (1 + 2 \cdot 3 + 2 \cdot (132 + 261)) \bmod x_0 = 131 \cdot 6 + (2 \cdot 3 + 1) = 793 \bmod 395 = 3 \\ &\qquad\qquad (\text{rumore} = 7 < p/2). \end{aligned}$$

La decodifica risulta corretta per entrambi:

$$\begin{aligned} m_1 &= (134 \bmod 131) \bmod 2 = 1, \\ m_2 &= (3 \bmod 131) \bmod 2 = 1. \end{aligned}$$

Proviamo a sommare e moltiplicare i testi cifrati, osservando che la decodifica è corretta:

$$\begin{aligned} c_1 + c_2 &= 134 + 3 = 137 \to (137 \bmod 131) \bmod 2 = 0 = m_1 + m_2 \\ c_1 \cdot c_2 &= 134 \cdot 3 = 402 \to (402 \bmod 131) \bmod 2 = 1 = m_1 \cdot m_2. \end{aligned}$$

1.2.3 Correttezza

Lemma 1.11. *Il crittosistema illustrato nella sezione precedente è corretto per $C_{\mathcal{E}}$.*

Dim. Consideriamo un testo cifrato appena ottenuto da Encrypt (nel seguito lo chiameremo "fresh") e suddividiamo la dimostrazione in due parti: la prima sostiene che ogni tale testo cifrato è vicino ad un multiplo di p e che la sua differenza dal più vicino multiplo di p ha la stessa parità di m; la seconda, invece, essenzialmente afferma che la stessa cosa è vera per i testi cifrati ottenuti dopo Evaluate.

1. *Sia* $(\mathsf{sk}, \mathsf{pk})$ *la coppia di chiavi data da* $\mathsf{KeyGen}(\lambda)$. *Sia* $c \xleftarrow{\mathcal{R}} \mathsf{Encrypt}(\mathsf{pk}, m)$ *per* $m \in \{0,1\}$. *Allora* $c = a \cdot p + (2b + m)$ *per qualche intero* a *e* b *con* $|2b + m| < 2^{\rho'+2}$.
   Per definizione $c \leftarrow (m + 2r + 2\sum_{i \in S} x_i) \bmod x_0$. Dato che $|x_0| > |x_i|$ per $i \in \{1, \ldots, \tau\}$, abbiamo

$$c = (m + 2r + 2\sum_{i \in S} x_i) + k \cdot x_0$$

   per qualche $|k| \leq \tau$, perché la parentesi fornisce interi lunghi al più $\tau\gamma$ bit e noi vogliamo ridurre tale valore modulo x_0, affinché risulti in $(-x_0/2, x_0/2]$; questo grazie alle relazioni in (1.1), che, ricordiamo, erano $\rho = \lambda$, $\rho' = 2\lambda$, $\eta \asymp \lambda^2$, $\gamma \asymp \lambda^5$, $\tau = \gamma + \lambda$.
   Per ogni i, esistono degli interi q_i e r_i con $|r_i| \leq 2^\rho$ tali che $x_i = q_i \cdot p + r_i$. Abbiamo

$$c = p \cdot (kq_0 + 2\sum_{i \in S} q_i) + (m + 2r + k \cdot 2\frac{r_0}{2} + \sum_{i \in S} 2r_i).$$

   Guardando il termine a destra, la sua parità è la stessa di m e il suo valore assoluto è al massimo $1 + 2^{\rho'+1} + \tau 2^\rho + \tau 2^{\rho+1} = 2^{\rho'}(\frac{1}{2^{\rho'}} + 2 + \frac{\tau}{2^\rho} + \frac{2\tau}{2^\rho}) < 2^{\rho'+2}$, dato che $\tau \ll 2^\rho$ ($\tau \asymp \lambda + \lambda^5$, $2^\rho = 2^\lambda$, i parametri dipendono da λ che è un valore adeguatamente grande, nel contesto possiamo considerare $\lambda = 72$).

2. *Sia* $(\mathsf{sk}, \mathsf{pk})$ *la coppia di chiavi data da* $\mathsf{KeyGen}(\lambda)$. *Sia* $C \in \mathcal{C}_\mathcal{E}$ *un circuito con* t *inputs e un output. Per* $i \in \{1, \ldots, t\}$ *e* $m_i \in \{0,1\}$, *sia* $c_i \xleftarrow{\mathcal{R}} \mathsf{Encrypt}(\mathsf{pk}, m_i)$. *Siano, inoltre,* $\mathfrak{m} \leftarrow C(m_1, \ldots, m_t)$ *e* $c \leftarrow \mathsf{Evaluate}(\mathsf{pk}, C, c_1, \ldots, c_t)$. *Allora* $c = a \cdot p + (2b + \mathfrak{m})$ *per qualche intero* a *e* b *con* $|2b + \mathfrak{m}| \leq p/8$.
   Sia C' il circuito generalizzato corrispondente a C, che opera sugli interi piuttosto che modulo 2. Generalmente abbiamo che $C'(c_1, \ldots, c_t) \in C'(2b_1 + m_1, \ldots, 2b_t + m_t) + p\mathbb{Z}$. Così $C'(2b_1 + m_1, \ldots, 2b_t + m_t) \bmod p$ ha la stessa parità di $\mathfrak{m} = C(m_1, \ldots, m_t)$. Abbiamo inoltre che $|C'(2b_1 + m_1, \ldots, 2b_t + m_t)| \leq 2^\eta/16 \leq p/8$ per definizione di $\mathcal{C}_\mathcal{E}$ (il valore assoluto dell'output è $\leq 2^{\eta-4}$, dato che $|2b_i + m_i| < 2^{\rho'+2}$ per il punto 1).

I punti 1 e 2 implicano immediatamente il lemma: per ogni circuito in $\mathcal{C}_\mathcal{E}$ e ogni cifratura degli input del circuito, l'intero ottenuto da $\mathsf{Evaluate}$ è della forma $c = a \cdot p + (2b + \mathfrak{m})$ con $|2b + \mathfrak{m}| \leq p/8$ (dove $\mathfrak{m}$ è il testo in chiaro che si suppone corrisponda al testo cifrato c). In tal modo abbiamo $c \bmod p = 2b + \mathfrak{m}$ e quindi $\mathfrak{m} = (c \bmod p) \bmod 2$. □

1.2.4 Sicurezza

Riduciamo la sicurezza del crittosistema "quasi" omomorfo precedentemente descritto alla difficoltà del problema del massimo comune divisore approssimato, meglio conosciuto come *approximate-gcd problem*, introdotto nel 2001 da Howgrave-Graham in [20]. Tale problema può essere così descritto: dato un insieme di interi $x_0, x_1, \ldots, x_\tau$, tutti scelti casualmente "vicini" a multipli di un intero grande p, trovare p.
Da ora in poi chiameremo "campioni" gli elementi (*samples*) presi casualmente dalla distribuzione $\mathcal{D}_{\gamma,\rho}(p)$.

Definizione 1.12. *Il (ρ, η, γ)-approximate-gcd problem è definito nel modo seguente: dato un numero polinomiale di campioni presi da $\mathcal{D}_{\gamma,\rho}(p) = \{x_i \ : \ x_i = pq_i + r_i,\ q_i \in \mathbb{Z} \cap [0, 2^\gamma/p), r_i \xleftarrow{\mathcal{R}} \mathbb{Z} \cap (-2^\rho, 2^\rho)\}$ per un intero dispari p composto da η bit scelto in modo casuale, trovare p.*

A differenza del massimo comun divisore, che può essere trovato tramite l'algoritmo di Euclide in un tempo polinomiale, l'approximate gcd è attualmente un problema computazionalmente difficile, richiedendo un numero esponenziale di operazioni (tramite una ricerca esaustiva del massimo comun divisore). Possiamo distinguere due versioni del problema a seconda che nella collezione esista o meno un multiplo esatto di p. Nel primo caso il problema è detto "speciale" (*partial approximate gcd*, PACD) e si richiede che il multiplo di p, $x_j = pq_j$, sia costituito con q_j molto grande in modo che x_j non sia facilmente fattorizzabile. Nel secondo caso il problema è detto "generale" (*general*, GACD). Attualmente in [6] Chen e Nguyen hanno mostrato che un attacco basato sul partial approximate gcd comporta un tempo d'esecuzione pari a numerosi anni.

Teorema 1.13. *Fissati i parametri $(\rho, \rho', \eta, \gamma, \tau)$ come all'inizio della sezione (tutti polinomiali nel parametro di sicurezza λ, si veda (1.1)), ogni attacco $\mathcal{A}$ con vantaggio ε sul crittosistema può essere convertito in un algoritmo $\mathcal{B}$ per risolvere il (ρ, η, γ)-approximate-gcd problem con probabilità di successo di almeno $\varepsilon/2$. Il tempo di esecuzione di $\mathcal{B}$ è polinomiale nel tempo di esecuzione di $\mathcal{A}$, in λ e in $1/\varepsilon$.*

Dim. Ricordiamo che $q_p(z)$ e $r_p(z)$ denotano il quoziente e il resto di z rispetto a p, quindi $z = q_p(z) \cdot p + r_p(z)$. Sia $\mathcal{A}$ un attacco contro il crittosistema; allora $\mathcal{A}$ prende come input una chiave pubblica e un testo cifrato (prodotti da KeyGen ed Encrypt del nostro schema) e fornisce come output il bit in chiaro corretto con probabilità $\frac{1}{2} + \varepsilon$ per qualche ε. (La probabilità è su

KeyGen ed Encrypt, come la scelta del bit in chiaro e l'interna casualità di $\mathcal{A}$.)
Usiamo $\mathcal{A}$ per costruire un algoritmo $\mathcal{B}$ per risolvere il problema dell'approximate-gcd con i parametri ρ, η, γ. Per una scelta casuale dell'intero dispari p di η bit, $\mathcal{B}$ ha accesso a tutti i campioni di $\mathcal{D}_{\gamma,\rho}(p)$ di cui ha bisogno; l'obiettivo è trovare p.

Step 1: Creare una chiave pubblica. $\mathcal{B}$ comincia costruendo una chiave pubblica per il crittosistema; per fare ciò estrae $\tau + 1$ campioni $x_0, \ldots, x_\tau$ da $\mathcal{D}_{\gamma,\rho}(p)$, li rietichetta in modo che x_0 sia il più grande e ricomincia tutta la procedura se x_0 non è dispari. Allora $\mathcal{B}$ dà come output la chiave pubblica $\mathsf{pk} = \langle x_0, x_1, \ldots, x_\tau \rangle$. Chiaramente, se $r_p(x_0)$ risulta pari, la distribuzione indotta sulla chiave pubblica è identica a quella dello schema.

Step 2: Una funzione per predire il bit meno significativo (LSB). Successivamente $\mathcal{B}$ produce una sequenza di interi e tenta di trovare p, utilizzando $\mathcal{A}$ per ottenere il bit meno significativo (*least-significant bit*) dei quozienti di questi interi rispetto a p. Ricordiamo che il bit meno significativo indica la posizione occupata dal bit nel sistema numerico binario che fornisce il valore dell'unità e che determina se il numero è pari o dispari; nella posizione convenzionale di una stringa di bit, quello meno significativo occupa l'ultimo posto a destra. $\mathcal{B}$ fa questo tramite la funzione che descriviamo mediante lo pseudocodice seguente:

Subroutine Learn $-$ $\mathsf{LSB}(z, \mathsf{pk})$:
Input: $z \in [0, 2^\gamma)$ con $|r_p(z)| < 2^\rho$, una chiave pubblica $\mathsf{pk} = \langle x_0, \ldots, x_\tau \rangle$
Output: il bit meno significativo di $q_p(z)$

1. For $j = 1$ to $\mathrm{poly}(\lambda)/\varepsilon$ do: //ε è il vantaggio totale di $\mathcal{A}$
2. Choose noise $r_j \xleftarrow{\mathcal{R}} (-2^{\rho'}, 2^{\rho'})$, a bit $m_j \xleftarrow{\mathcal{R}} \{0, 1\}$ and a random subset $S_j \subseteq_R \{1, \ldots, \tau\}$
3. Set $c_j \leftarrow (z + m_j + 2r_j + 2\sum_{k \in S_j} x_k) \bmod x_0$
4. Call $\mathcal{A}$ to get a prediction $a_j \leftarrow \mathcal{A}(\mathsf{pk}, c_j)$
5. Set $b_j \leftarrow a_j \oplus \mathsf{parity}(z) \oplus m_j$ // b_j dovrebbe essere la parità di $q_p(z)$
6. Output the majority vote among the b_j's.

Nel Lemma 1.14 mostreremo che, per tutte tranne una trascurabile frazione delle chiavi pubbliche generate dal crittosistema, il testo cifrato c_j della riga 3 dello pseudocodice

precedente è estrapolato quasi in modo identico ad una codifica valida del bit $r_p(z)$ mod $2 \oplus m_j$. Inoltre, dato che p è dispari, possiamo notare che $q_p(z)$ mod $2 = r_p(z)$ mod $2 \oplus \mathsf{parity}(z)$. Segue che, se $\mathcal{A}$ ha un vantaggio notevole nell'indovinare il bit cifrato sotto pk, allora $\mathsf{Learn} - \mathsf{LSB}(z, \mathsf{pk})$ fornirà $q_p(z)$mod 2 con probabilità molto vicina a 1.

Step 3: Binary GCD. La procedura dello Step 2 permette di interpretare l'azione di $\mathcal{A}$ come quella di un "oracolo" utile ad ottenere il bit meno significativo di $q_p(z)$. Fatto questo, siamo molto vicini a trovare p. Una strada conveniente per attuare ciò è utilizzare l'algoritmo del *binary GCD*: dati due interi qualsiasi $z_1 = q_p(z_1) \cdot p + r_p(z_1)$ e $z_2 = q_p(z_2) \cdot p + r_p(z_2)$ (con $r_p(z_i) \ll p$, $i = 1, 2$), applicare loro il seguente processo ripetutamente:

1. se $z_2 > z_1$ allora scambiare $z_1 \leftrightarrow z_2$;
2. usare un oracolo per conoscere il bit di parità di $q_p(z_1)$ e $q_p(z_2)$, sia esso $b_i = q_p(z_i)$ mod 2;
3. se entrambi i $q_p(z_i)$ sono dispari allora sostituire z_1 con $z_1 \leftarrow z_1 - z_2$ e porre $b_1 \leftarrow 0$;
4. per ogni z_i con $b_i = 0$ sostituire z_i con $z_i \leftarrow (z_i - \mathsf{parity}(z_i))/2$ (notiamo che $(z_i - \mathsf{parity}(z_i))$ è pari, quindi il nuovo z_i è ancora un intero).

Osserviamo che, quando $p \gg r_p(z_i)$, sottrarre il bit di parità non cambia il quoziente fatto rispetto a p, ma cambia solo il resto: $q_p(z_i - \mathsf{parity}(z_i)) = q_p(z_i)$. Segue che, posto $z_i' \leftarrow (z_i - \mathsf{parity}(z_i))/2$ nella riga 4 (in cui sappiamo che $q_p(z_i)$ è pari), otteniamo

$$q_p(z_i') = q_p(z_i)/2 \text{ e } r_p(z_i') = (r_p(z_i) - \mathsf{parity}(z_i))/2.$$

Mostriamo ora che il rumore in z_1 e in z_2 non "cresce mai troppo" in questo processo. Chiaramente, ponendo $z_i' \leftarrow (z_i - \mathsf{parity}(z_i))/2$ nella riga 4 , abbiamo $|r_p(z_i')| \leq (|r_p(z_i)| + 1)/2 \leq |r_p(z_i)|$; inoltre, quando sostituiamo z_1 con $z_1 \leftarrow z_1 - z_2$ nella riga 3 e poi con $z_i'' \leftarrow (z_1' - \mathsf{parity}(z_i'))/2$ nella riga 4, risulta

$$|r_p(z_1'')| = \frac{|r_p(z_1') - \mathsf{parity}(z_1')|}{2} = \frac{|r_p(z_1) - r_p(z_2) - \mathsf{parity}(z_1')|}{2} \leq \max\{|r_p(z_1)|; |r_p(z_2)|\}.$$

Quindi i resti $r_p(z_i)$ non crescono mai oltre i due iniziali, così $p \gg r_p(z_i)$. Questo implica che le operazioni precedenti corrispondono alle operazioni usuali dell'algoritmo del binary GCD applicato ai quozienti $q_p(z_i)$. Dopo $O(\gamma)$ iterazioni (γ è il numero di bit degli z_i) avremo i due interi z_1' e z_2' con $z_2' = 0$ e $q_p(z_1')$ la parte dispari del $GCD(q_p(z_1), q_p(z_2))$ per i due interi iniziali.

Step 4: Trovare p. Per trovare p l'algoritmo $\mathcal{B}$ preleva una coppia di elementi $z_1^*, z_2^* \xleftarrow{\mathcal{R}} \mathcal{D}_{\gamma,\rho}(p)$ e applica loro l'algoritmo del binary GCD. Con probabilità di almeno $6/\pi^2 \approx 0.6$ [1], la parte dispari del $GCD(q_p(z_1^*), q_p(z_2^*))$ è 1 e ciò significa che la procedura darà come output un elemento $\widetilde{z} = 1 \cdot p + r$ con $|r| \leq 2^\rho$. Se questo non succede, allora $\mathcal{B}$ estrae due nuovi interi e prova ancora.

Infine $\mathcal{B}$ ripete la procedura del binary GCD suddetta usando $z_1 = z_1^*$ e $z_2 = \widetilde{z}$; la sequenza dei bit di parità dei $q_p(z_1)$ in tutte le iterazioni esprime la rappresentazione di $q_p(z_1^*)$. Ora $\mathcal{B}$ recupera $p = \lfloor z_1^*/q_p(z_1^*) \rceil$.

Riepilogo: Abbiamo mostrato che $\mathcal{B}$ è in grado di trovare p avendo accesso ad un oracolo attendibile per calcolare $q_p(z)$ mod 2 (per gli z con rumore molto più piccolo di p). Manca da analizzare la probabilità (sulla scelta di $\mathcal{B}$ della chiave pubblica) che ci permetta di concludere che la procedura $\mathsf{Learn-LSB}(z, \mathsf{pk})$ sia davvero un oracolo affidabile.

La probabilità di successo di $\mathcal{B}$: Di seguito proveremo una semplice tecnica sulla distribuzione dei testi cifrati nel nostro crittosistema. Ricordiamo che la distribuzione della chiave pubblica che $\mathcal{B}$ genera è identica alla distribuzione corretta del crittosistema, condizionata a qualche evento di probabilità uguale a $\frac{1}{2}$ (i.e. $q_p(x_0)$ è dispari). Chiamiamo $\mathcal{G}$ tale "buon" evento. Come già accennato, nel Lemma 1.14 proveremo che, per ogni chiave segreta p e per tutte tranne una frazione trascurabile di chiavi pubbliche (generate da KeyGen), la procedura che $\mathcal{B}$ usa per generare testi cifrati nella riga 3 della funzione $\mathsf{Learn-LSB}$ genera una distribuzione che è statisticamente vicina alla distribuzione di testi cifrati del crittosistema. Questo ci permette di analizzare la probabilità di successo di $\mathcal{B}$ come segue. Sia $\mathcal{P}$ l'insieme degli interi in $[2^{\eta-1}, 2^\eta)$ per il quale $\mathcal{A}$ ha un vantaggio maggiore di $\varepsilon/2$:

$$\mathcal{P} := \{p \in [2^{\eta-1}, 2^\eta) \ : \ \mathsf{advantage}(\mathcal{A}) \text{ condizionato a } \mathsf{sk} = p \text{ sia almeno } \varepsilon/2\};$$

pertanto la probabilità di avere degli interi dispari da $[2^{\eta-1}, 2^\eta)$ che stiano in $\mathcal{P}$ è almeno $\varepsilon/2$.

Analogamente, per un dato $p \in \mathcal{P}$, denotiamo con $\mathcal{PK}_p$ l'insieme di chiavi pubbliche per le quali $\mathcal{A}$ ha vantaggio di almeno $\varepsilon/4$:

$$\mathcal{PK}_p := \{\mathsf{pk} \text{ per } p \ : \ \mathsf{advantage}(\mathcal{A}) \text{ condizionato a } \mathsf{pk} = p \text{ sia almeno } \varepsilon/4\};$$

[1] Ricordiamo che la probabilità che, scelti casualmente due interi, essi risultino coprimi è $1/\zeta(2)$, dove $\zeta(2) = \sum_{n=1}^{\infty} n^{-2} = \pi^2/6$ (si veda l'Appendice A).

di nuovo, per ogni $p \in \mathcal{P}$, l'algoritmo KeyGen (quando usiamo la chiave segreta $\mathsf{sk} = p$) deve fornire $\mathsf{pk} \in \mathcal{PK}_p$ con probabilità di almeno $\varepsilon/4$.

Consideriamo ora una singola esecuzione di $\mathcal{B}$ quando campiona $\mathcal{D}_{\gamma,\rho}(p)$ per qualche $p \in \mathcal{P}$. Se con probabilità $1/2$ l'evento $\mathcal{G}$ accade, in tal caso la chiave pubblica che $\mathcal{B}$ produce è trascurabilmente vicina alla distribuzione corretta. Quindi, condizionato a $\mathcal{G}$, $\mathcal{B}$ genera qualche $\mathsf{pk} \in \mathcal{PK}_p$ con probabilità $\varepsilon' \geq \varepsilon/4 - \mathsf{negl}$, dove negl indica un termine trascurabile (negligible). Inoltre, per il Lemma 1.14, con probabilità $\varepsilon' - \mathsf{negl}$ non solo la chiave pubblica sta in $\mathcal{PK}_p$, ma anche la generazione di testi cifrati che $\mathcal{B}$ usa nella riga 3 di $\mathsf{Learn} - \mathsf{LSB}$ utilizza questa chiave pubblica (ciò significa che i testi cifrati che genera sono scelti quasi dalla distribuzione corretta). Se accade questo, allora $\mathcal{A}$ fornisce la risposta giusta nella riga 4 di $\mathsf{Learn} - \mathsf{LSB}$ con probabilità $\varepsilon/4 - \mathsf{negl}$. Quando la funzione chiama $\mathcal{A}$ e prende il risultato più ricorrente, si otterrà la risposta corretta con probabilità molto vicina a 1 e $\mathcal{B}$ troverà il massimo comune divisore approssimato p.

Quando la chiave segreta è $p \in \mathcal{P}$ allora $\mathcal{B}$ ha la probabilità di almeno $1/2 \cdot (\varepsilon/4 - \mathsf{negl})$ di trovarlo in una singola esecuzione. Se $\mathcal{B}$ chiama l'oracolo $(8/\varepsilon) \cdot \omega(\log \lambda)$ volte e $\mathcal{A}$ si può ripetere per $\mathrm{poly}(\lambda)/\varepsilon$ volte, abbiamo un algoritmo di complessità $\mathrm{poly}(\lambda, 1/\varepsilon)$ che lavora per ogni $p \in \mathcal{P}$; così la totale probabilità di successo di questo risolutore è almeno la densità di $\mathcal{P}$, ovvero $\varepsilon/2$. Questo completa la dimostrazione. □

Lemma 1.14. *Fissiamo i parametri $(\rho, \rho', \eta, \gamma, \tau)$, una chiave segreta $\mathsf{sk} = p$ e sia la chiave pubblica $\mathsf{pk} = \langle x_0, \ldots, x_\tau \rangle$ scelta casualmente come in KeyGen del nostro crittosistema. Per ogni intero $x^* \in [0, 2^\gamma]$ che sia lontano al massimo 2^ρ da un multiplo di p, consideriamo la distribuzione $\mathcal{C}_{\mathsf{pk}}(x^*)$, in cui sono raccolti i risultati ottenuti dalla seguente procedura:*

- $S \subseteq_R \{1, \ldots, \tau\}$,
- $r \xleftarrow{\mathcal{R}} (-2^{\rho'}, 2^{\rho'})$,
- $\mathsf{output} : c' \leftarrow (x^* + 2r + \sum_{i \in S} x_i) \bmod x_0$.

Allora con probabilità molto vicina ad 1 (sulla scelta di $(\mathsf{sk}, \mathsf{pk})$), ogni distribuzione $\mathcal{C}_{\mathsf{pk}}(x^*)$ è statisticamente vicina alla distribuzione $\mathsf{Encrypt}(\mathsf{pk}, m = x^* \bmod 2)$ (fino a una trascurabile distanza statistica).

Dim. Scrivendo $c' = q'p + 2r' + m$ e controlliamo separatamente q' e r'. Per quanto riguarda q' affermiamo che $q_p(c)$ di un testo cifrato è uniforme in $(-q_0/2, q_0/2]$ per il Leftover Hash Lemma; dato che la somma usata per generare c' è come nel crittosistema, questo implica che il valore q' è uniforme anche in $(-q_0/2, q_0/2]$ e la nostra asserzione segue dal Lemma 1.15 seguente. Per applicare il lemma in modo significativo (ossia per assicurarci che la distribuzione sia essenzialmente uniforme) abbiamo bisogno che $\tau > \log q_0 + \omega(\log \lambda)$, come indicato nella nostra scelta di parametri (1.1).
Per quanto riguarda il rumore r', ricordiamo che abbiamo scelto del rumore addizionale per il testo cifrato sia nel crittosistema sia nella riduzione, estratto da una distribuzione con ampiezza maggiore rispetto al rumore degli elementi della chiave pubblica (di un fattore superpolinomiale). Questo rumore aggiunto diventa statisticamente trascurabile qualsiasi differenza esista nella distribuzione del rumore dovuto agli elementi della chiave pubblica e all'intero x^*.
□

Lemma 1.15. *Sia $T > 0$ un intero. Consideriamo la seguente distribuzione: poniamo $x_1, \ldots, x_T \xleftarrow{\mathcal{R}} \mathbb{Z}_M$ uniformemente e indipendentemente, $\boldsymbol{s} \xleftarrow{\mathcal{R}} \{0,1\}^T$ e $x_{T+1} \leftarrow \sum_{i=1}^{T} s_i \cdot x_i \bmod M$, allora $(x_1, \ldots, x_T, x_{T+1})$ è $\frac{1}{2}\sqrt{M/2^T}$-uniforme su $\mathbb{Z}_M^{T+1}$.*

Dim. Definiamo una famiglia $\mathcal{H}$ di funzioni hash da $\{0,1\}^T$ a $\mathbb{Z}_M$ come segue: i membri $h \in \mathcal{H}$ sono associati agli elementi $(h_1, \ldots, h_T) \in \mathbb{Z}_M^T$ e, per $\mathbf{s} \in \{0,1\}^T$, $h(\mathbf{s})$ è data da $\sum_{i=1}^{T} s_i \cdot h_i \in Z_M$. Questa famiglia è 2-universale, allora, per il Leftover Hash Lemma (Lemma 1.16), $(h, h(x))$ è $\frac{1}{2}\sqrt{M/2^T}$-uniforme su $\mathbb{Z}_M^{T+1}$. □

Abbiamo usato la seguente versione semplificata del Leftover Hash Lemma.

Lemma 1.16 (Simplified Leftover Hash Lemma)**.** *Sia $\mathcal{H}$ una famiglia di funzioni hash 2-universali da X a Y. Supponiamo che $h \xleftarrow{\mathcal{R}} \mathcal{H}$ e $x \xleftarrow{\mathcal{R}} X$ siano scelti uniformemente e indipendentemente. Allora $(h, h(x))$ è $\frac{1}{2}\sqrt{|Y|/|X|}$-uniforme su $\mathcal{H} \times Y$.*

1.2.5 Leftover Hash Lemma

Per dimostrare il Leftover Hash Lemma seguiamo l'impostazione che troviamo nel libro di Shoup [28]. Innanzitutto definiamo due utili misure di casualità, a partire da una variabile aleatoria X a valori in S, insieme di cardinalità m:

- la **probabilità di collisione** di X è $\sum_{s \in S}(P[X = s])^2$;

- la **distanza di X dalla distribuzione uniforme su** S è $\frac{1}{2}\sum_{s\in S}|P[X=s]-1/m|$.

Osservazione 1.17. *Siano $\alpha_1,\dots,\alpha_m$ numeri reali tali che $\sum_{s=1}^m \alpha_s = 1$, allora*

$$0 \le \sum_{s=1}^{m}\Big(\alpha_s - \frac{1}{m}\Big)^2 = \sum_{s=1}^{m}\Big(\alpha_s^2 - \frac{2\alpha_s}{m} + \frac{1}{m^2}\Big) = \sum_{s=1}^{m}\alpha_s^2 - \frac{2}{m}\cdot 1 + \frac{m}{m^2} = \sum_{s=1}^{m}\alpha_s^2 - \frac{1}{m}.$$

In particolare $\sum_{s=1}^m \alpha_s^2 \ge 1/m$.

Definizione 1.18. *Sia $\{\Phi_r\}_{r\in R}$ una famiglia di funzioni hash da S a T, con $|T|=m$, e sia H una variabile aleatoria distribuita uniformemente su R. Diciamo che $\{\Phi_r\}_{r\in R}$ è ε-**quasi universale** se per ogni $s,s'\in S$ con $s\neq s'$, abbiamo $P[\Phi_H(s)=\Phi_H(s')]\le\varepsilon$.*

Teorema 1.19. *Supponiamo che X sia una variabile aleatoria a valori in un insieme finito S di cardinalità m. Se X ha probabilità di collisione β e distanza δ dalla distribuzione uniforme su S, allora $\delta \le \frac{1}{2}\sqrt{m\beta-1}$.*

Dimostrazione. Per $s\in S$, sia $p_s := P[X=s]$, segue allora dalla definizione di distanza statistica che $\delta = \frac{1}{2}\sum_{s\in S}|p_s-1/m|$ e si ha $1=\sum_s q_s$, dove $|p_s-1/m|/2\delta$. Quindi, per l'Osservazione 1.17,

$$\frac{1}{m} \le \sum q_s^2 = \frac{1}{4\delta^2}\sum_s (p_s-1/m)^2 = \frac{1}{4\delta^2}\Big(\sum_s p_s^2 - 1/m\Big) = \frac{1}{4\delta^2}(\beta-1/m),$$

da cui segue immediatamente la tesi: $\delta \le \frac{1}{2}\sqrt{m\beta-1}$. □

Teorema 1.20. (**Leftover Hash Lemma**) *Sia $\{\Phi_r\}_{r\in R}$ una famiglia di funzioni hash $(1+\alpha)/m$-quasi universale da S a T, dove $m=|T|$. Siano H e X due variabili casuali indipendenti, sia inoltre H distribuita uniformemente su R e X sia a valori in S. Se β è la probabilità di collisione di X e δ' è la distanza di $(H,\Phi_H(X))$ dalla distribuzione uniforme su $R\times T$, allora $\delta' \le \sqrt{m\beta+\alpha}$.*

Dimostrazione. Sia β' la probabilità di collisione di $(H,\Phi_H(X))$, il nostro obiettivo è limitare β' e poi applicare il Teorema 1.19 alla variabile aleatoria $(H,\Phi_H(X))$. Per fare ciò poniamo $l := |R|$ e supponiamo che H' e X' siano variabili casuali tali che H' abbia la stessa distribuzione di H, X' abbia la stessa distribuzione di X e H,H',X,X' formino una famiglia mutuamente

indipendente di variabili aleatorie. Allora abbiamo

$$\begin{aligned}
\beta' &= P[(H = H') \cap (\Phi_H(X) = \Phi_{H'}(X'))] = P[(H = H') \cap (\Phi_H(X) = \Phi_H(X'))] \\
&= \frac{1}{l} P[\Phi_H(X) = \Phi_H(X')] \qquad \text{perché sono indipendenti e } H \text{ è uniforme su } R \\
&\leq \frac{1}{l}\big(P[X = X'] + (1+\alpha)/m\big) \text{ perché } \{\Phi_r\}_{r\in R} \text{ è } (1+\alpha)/m\text{-quasi universale} \\
&= \frac{1}{lm}(m\beta + 1 + \alpha).
\end{aligned}$$

Il teorema segue pertanto dal Teorema 1.19. □

1.3 Rendere il crittosistema pienamente omomorfo

Seguiamo l'approccio di Gentry e costruiamo un crittosistema pienamente omomorfo a partire da uno schema parzialmente omomorfo che sia reso bootstrappable tramite una trasformazione per comprimere ("squash") il circuito di decodifica. In questa trasformazione aggiungiamo alla chiave pubblica qualche informazione in più sulla chiave segreta e usiamo queste informazioni extra per post-processare il testo cifrato, che può essere decifrato in modo più efficiente del testo cifrato originale.
Precisiamo che di seguito utilizzeremo la notazione $z \bmod p$ a intendere $z - \lfloor z/p \rceil \cdot p$.

1.3.1 "Compressione" del circuito di decodifica (Squashing the decryption circuit)

Fissiamo tre ulteriori parametri, κ, θ e Θ, che siano funzioni del parametro di sicurezza λ. Di seguito useremo $\kappa = \gamma\eta/\rho'$, $\theta = \lambda$ e $\Theta = \omega(\kappa \cdot \log\lambda)$. Per una chiave segreta $\mathsf{sk}^* = p$ e una chiave pubblica pk^* dal nostro schema originale parzialmente omomorfo $\mathcal{E}^*$, aggiungiamo alla chiave pubblica un insieme $\mathbf{y} = \{y_1, \ldots, y_\Theta\}$ di numeri razionali in $[0, 2)$ con κ bit di precisione, tale che ci sia un sottoinsieme $S \subset \{1, \ldots, \Theta\}$ di cardinalità θ con $\sum_{i\in S} y_i \approx 1/p \bmod 2$. Modifichiamo il crittosistema descritto precedentemente in §1.2.2 come segue:

KeyGen. Genera $\mathsf{sk}^* = p$ e pk^* come prima, fissa $x_p \leftarrow \lfloor 2^\kappa/p \rceil$, sceglie casualmente un vettore di Θ bit con peso di Hamming θ (che indica il numero di bit diversi da zero nella stringa e coincide con la somma dei bit), $\mathbf{s} = (s_1, \ldots, s_\Theta)$ e sia $S = \{i : s_i = 1\}$.
Sceglie casualmente gli interi $u_i \in \mathbb{Z} \cap [0, 2^{\kappa+1})$, per $i = 1, \ldots, \Theta$, soggetti alla condizione che

$\sum_{i\in S} u_i = x_p \bmod 2^{\kappa+1}$. Pone $y_i = u_i/2^\kappa$ e $\mathbf{y} = \{y_1, \ldots, y_\Theta\}$, ogni y_i è un numero positivo minore di 2 con κ bit di precisione dopo la virgola binaria, inoltre $\left(\sum_{i\in S} y_i\right) \bmod 2 = (1/p) - \Delta_p$ per qualche $\Delta_p < 2^{-\kappa}$.
L'output fornito è la chiave segreta $\mathsf{sk} = \mathbf{s}$ e la chiave pubblica $\mathsf{pk} = (\mathsf{pk}^*, \mathbf{y})$.

Encrypt e Evaluate. Genera un testo cifrato c^*, un intero come in precedenza, $c^* = (m + 2r + 2\sum x_i) \bmod x_0$; dopodiché per $i \in \{1, \ldots, \Theta\}$ pone $z_i \leftarrow (c^* \cdot y_i) \bmod 2$, tenendo solo $n = \lceil \log \theta \rceil + 3$ bit di precisione dopo la virgola binaria per ogni z_i. Fornisce in output c^* e $\mathbf{z} = (z_1, \ldots, z_\Theta)$.

Decrypt. Dà come risultato $m' \leftarrow \left(c^* - \left\lfloor \sum_i s_i z_i \right\rceil\right) \bmod 2$.

Mostriamo che il crittosistema, nonostante sia stato modificato, rimane corretto per i circuiti permessi.

Lemma 1.21. *Lo schema modificato sopra descritto è corretto per ogni circuito permesso C in $\mathcal{C}_\mathcal{E}$. Inoltre, per ogni testo cifrato $(c^*, \mathbf{z})$ generato da* Evaluate*, si ha che $\sum s_i z_i$ dista al più $1/4$ da un intero.*

Dimostrazione. Fissiamo una chiave pubblica e una segreta generate rispetto al parametro di sicurezza λ, con i numeri razionali $\{y_i\}_{i=1}^\Theta$ nella chiave pubblica e i bit $\{s_i\}_{i=1}^\Theta$ della chiave segreta; ricordiamo che le y_i sono state scelte in modo tale che $\left(\sum_i s_i y_i\right) \bmod 2 = (1/p) - \Delta_p$ con $|\Delta_p| \leq 2^{-\kappa}$.
Fissiamo un polinomio permesso $P(x_1, \ldots, x_t)$, un circuito C che calcoli P e t testi cifrati $\{c_i\}_{i=1}^t$ e denotiamo con c^* l'output di Evaluate, $c^* = \mathsf{Evaluate}(\mathsf{pk}, C, c_1, \ldots, c_t)$. Dobbiamo mostrare che

$$\lfloor c^*/p \rceil = \left\lfloor \sum_i s_i z_i \right\rceil \bmod 2,$$

dove gli z_i sono calcolati come $(c^* \cdot y_i) \bmod 2$ con solo $\lceil \log \theta \rceil + 3$ bit di precisione dopo la

virgola binaria, così si ha che $(c^* \cdot y_i) \bmod 2 = z_i - \Delta_i$ con $|\Delta_i| \leq \frac{1}{16\theta}$. Ne consegue

$$\begin{aligned}
\Big(c^*/p - \sum_i s_i z_i\Big) \bmod 2 &= \Big(c^*/p - \sum_i s_i((c^* \cdot y_i) \bmod 2) + \sum_i s_i \Delta_i\Big) \bmod 2 \\
&= \Big(c^*/p - c^* \cdot \Big(\Big(\sum_i s_i y_i\Big) \bmod 2\Big) + \sum_i s_i \Delta_i\Big) \bmod 2 \\
&= \Big(c^*/p - c^* \cdot (1/p - \Delta_p) + \sum_i s_i \Delta_i\Big) \bmod 2 \\
&= \Big(c^* \cdot \Delta_p + \sum_i s_i \Delta_i\Big) \bmod 2.
\end{aligned}$$

Noi asseriamo che la quantità finale dentro parentesi ha valore assoluto pari al massimo a 1/8. Per definizione, dato che c^* è un testo cifrato valido risultante da un polinomio permesso, il valore c^* dista al più 1/8 da un intero. Per dimostrare quanto asserito, osserviamo che $\big|\sum s_i \Delta_i\big| \leq \theta \cdot \frac{1}{16\theta} = 1/16$. Per quanto riguarda $c^* \cdot \Delta_p$, ricordiamo che il testo cifrato c^* è ottenuto valutando il polinomio P sui testi cifrati c_i. Per definizione di polinomio permesso, per ogni $\alpha \geq 1$, se gli input di P hanno valore assoluto al massimo pari a $2^{\alpha(\rho'+2)}$, l'output sarà al massimo $2^{\alpha(\eta-4)}$. In particolare quando gli input di P sono "fresh" hanno valore al massimo 2^γ; inoltre c^* risultante da P ha valore al massimo $2^{\gamma(\eta-4)/(\rho'+2)} < 2^{\kappa-4}$. Essendo $|\Delta_p| < 2^{-\kappa}$, si ottiene $|c^* \cdot \Delta_p| < 1/16$. Quindi $|c^* \cdot \Delta_p + \sum_i s_i \Delta_i| < 1/16 + 1/16 = 1/8$, da cui consegue $\lfloor c^*/p \rceil = \lfloor \sum_i s_i z_i \rceil \bmod 2$; pertanto il sistema modificato risulta corretto.
Per quanto riguarda la seconda affermazione dell'enunciato, essendo $z_i \leftarrow (c^* \cdot y_i) \bmod 2$ e sapendo che c^* dista 1/8 da un intero e $y_i \in [0, 2)$, la somma $\sum s_i z_i$ dista al più $\frac{1}{8} \cdot 2 = \frac{1}{4}$ da un intero. □

1.3.2 Il crittosistema è bootstrappable

Possiamo ora mostrare che, a seguito delle modifiche effettuate, il crittosistema è diventato bootstrappable e, quindi, è in grado di "pulire" i testi cifrati, che a un certo punto risultano affetti da un eccessivo rumore dovuto alle varie operazioni svolte in Evaluate. Una volta che il rumore in eccesso viene rimosso, si può proseguire con ulteriori operazioni e così si ottiene un crittosistema pienamente omomorfo.

Teorema 1.22. *Sia $\mathcal{E}$ il crittosistema sopra descritto e sia $D_\mathcal{E}$ l'insieme dei circuiti di decifratura aumentati (ottenuti dopo la compressione), allora $D_\mathcal{E} \subset \mathcal{C}_\mathcal{E}$, in altre parole $\mathcal{E}$ è bootstrappable.*

Dimostrazione. L'obiettivo consiste nell'esprimere l'equazione della decodifica modificata

$$m' \leftarrow c^* - \left\lfloor \sum s_i z_i \right\rceil \bmod 2$$

come un polinomio permesso f, soddisfacente la relazione (1.2), $d \leq \frac{\eta - 4 - \log|\vec{f}|}{\rho' + 2}$, dove $|\vec{f}|$ è la norma l_1 del vettore dei coefficienti $\vec{f}$, e mostriamo che c'è un circuito polinomiale che lo computi. Ricordiamo che c^* è un intero, gli s_i sono bit e gli z_i sono numeri razionali in $[0, 2)$ con rappresentazione binaria avente $n = \lceil \log \theta \rceil + 3$ bit di precisione dopo la virgola binaria; inoltre sappiamo che $\sum s_i z_i$ dista al più $1/4$ da un intero e che solo θ dei bit $s_1, \ldots, s_\Theta$ sono diversi da zero.
Suddividiamo il calcolo in tre fasi:

1. Per $i \in \{1, \ldots, \Theta\}$ poniamo $a_i \leftarrow s_i z_i$, ovvero $a_i = z_i$ quando $s_i = 1$ e $a_i = 0$ altrimenti. Gli a_i sono numeri razionali in $[0, 2)$ dati in rappresentazione binaria con n bit di precisione dopo la virgola binaria.

2. Dai Θ numeri razionali $\{a_i\}_{i=1}^{\Theta}$ generiamo altri $n+1$ razionali $\{w_j\}_{j=0}^{n}$, ognuno con meno di n bit di precisione, tali che $\sum_j w_j = \sum_i a_i \bmod 2$.

3. Diamo come risultato $\left(c^* - \sum_j w_j\right) \bmod 2$.

Il primo step può essere ottenuto con un sotto-circuito di livello 1 di porte moltiplicative (è richiesta una sola moltiplicazione), mentre per la seconda e terza fase sono richiesti sotto-circuiti più complessi.
In generale per calcolare la somma di k numeri razionali r_i in rappresentazione binaria, $\sum_{i=1}^{k} r_i$, abbiamo delle tecniche conosciute, tra cui il "three-for-two trick", per il quale un circuito di profondità costante è usato per trasformare tre numeri di lunghezza arbitraria in due numeri che sono al massimo più lunghi di un bit, tali che la somma dei due numeri risultanti sia la stessa della somma dei tre input iniziali. (I bit di output del circuito sono espressioni lineari o quadratiche con tre monomi nei bit di input.) Applicando questo stratagemma al massimo $\lceil \log_{3/2} k \rceil + 2$ volte si ottengono due numeri $s_1 + s_2 = \sum_{i=1}^{k} r_i$. Pertanto la profondità totale necessaria per ridurre k numeri in due numeri è $d' \leq 2^{\lceil \log_{3/2} k \rceil + 2} < 8k^{1/\log(3/2)} < 8k^{1.71}$. La profondità del circuito necessario per calcolare la somma finale di due numeri è logaritmica nel loro numero di bit, ma siamo interessati a $\lfloor s_1 + s_2 \rceil \bmod 2$ e sappiamo che $s_1 + s_2$ dista $1/4$ da un intero, ciò può essere calcolato da un polinomio in più variabili di quarto grado con solo

nove termini. Quindi il circuito che calcola $\lfloor \sum_{i=1}^{k} r_i \rceil \bmod 2$ corrisponde ad un polinomio di grado al massimo $d \leq 32k^{1/\log(3/2)}$ con vettore dei coefficienti avente norma l_1 al massimo pari a 27^d. Sfortunatamente questo grado (con $k = \Theta$) è ancora troppo grande, quindi usiamo la tecnica di Gentry che sfrutta il fatto che tutti tranne θ degli elementi a_i sono nulli.
Denotiamo la rappresentazione binaria di ogni numero a_i con

$$a_{i,0\bullet}a_{i,-1}a_{i,-2}\ldots a_{i,-n}, \quad a_i = \sum_{j=0}^{n} 2^{-j}a_{i,-j}.$$

Il cuore della procedura è una funzione per calcolare interi W_{-j} per $j = 0, \ldots, n$, dove W_{-j} è il peso di Hamming della colonna di bit $(a_{1,-j}, a_{2,-j}, \ldots, a_{\Theta,-j})$, come esemplificato dalla figura di seguito:

$$\begin{array}{ccccc} a_{1,0} & a_{1,-1} & a_{1,-2} & \ldots & a_{1,-n} \\ a_{2,0} & a_{2,-1} & a_{2,-2} & \ldots & a_{2,-n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{\Theta,0} & a_{\Theta,-1} & a_{\Theta,-2} & \ldots & a_{\Theta,-n} \\ \hline W_0 & W_{-1} & W_{-2} & \ldots & W_{-n} \end{array}$$

Dato che al massimo θ degli a_i sono non nulli allora i W_{-j} non sono maggiori di θ e perciò possono essere rappresentati da $\lceil \log(\theta + 1) \rceil < n$ bit. Dal lemma seguente, ogni bit nella rappresentazione binaria di W_{-j} può essere espresso da un polinomio di grado al massimo θ nelle Θ variabili $a_{i,-j}$, per $i = 1, 2, \ldots, \Theta$. Inoltre tutti questi polinomi possono essere calcolati simultaneamente da un circuito aritmetico di dimensione $O(\theta \cdot \Theta)$.
Una volta ottenuti i pesi di Hamming W_{-j}, la somma degli a_i può essere ottenuta da $\sum_i a_i = \sum_j 2^{-j}W_{-j}$. Per $j = 0, 1, \ldots, n$ poniamo $w_j = (2^{-j} \cdot W_{-j}) \bmod 2$, così i w_j sono numeri razionali con $\lceil \log(\theta+1) \rceil < n$ bit di precisione. Ora possiamo sommare i w_j usando il three-for-two trick questa volta con $k = n + 1$ e otteniamo la somma degli $a_i \bmod 2$. Concludiamo che il grado del polinomio nella prima fase è due, il grado nella seconda fase è al massimo θ e il grado nella terza fase è al massimo

$$32(n+1)^{1/\log(3/2)} < 32\lceil \log\theta + 4 \rceil^{1.71} < 32\log^2\theta.$$

Quindi il grado totale del circuito di decodifica è limitato da $2 \cdot \theta \cdot 32\log^2\theta = 64\theta\log^2\theta$ e dato che $\theta = \lambda$ il grado diventa al massimo $64\lambda\log^2\lambda$. Segue che i circuiti di decodifica aumentati $D_{\mathcal{E}}$ (i.e. la decodifica è seguita da una singola moltiplicazione o addizione) possono essere espressi come polinomi di grado al massimo $128\lambda\log^2\lambda$ nelle Θ variabili s_i. Dato che il logaritmo della norma l_1 di questo polinomio è piccola rispetto a η e dato che $\Theta = \frac{\eta\gamma}{\rho} \cdot \omega(\log\lambda) < \lambda^7$ (e anche

$\tau < \lambda^7$), l'osservazione 1.9 (con $\alpha = 128$ e $\beta = 7$) indica che $D_{\mathcal{E}} \subset \mathcal{C}(\mathcal{P}_{\mathcal{E}})$), rendendo il sistema bootstrappable se $\eta = \rho \cdot \Theta(\lambda \log^2 \lambda)$. □

Dobbiamo ancora mostrare come calcolare i pesi di Hamming W_j usando un polinomio di grado non superiore a θ.

Lemma 1.23. *Sia $\vec{\sigma} = (\sigma_1, \ldots, \sigma_t)$ un vettore binario e sia $W = W(\vec{\sigma})$ il peso di Hamming di $\vec{\sigma}$ ed esprimiamo la rappresentazione binaria di W con $W_n \ldots W_1 W_0$, ovvero $W = \sum_{i=0}^{n} 2^i W_i$ e tutti i W_i sono bit. Allora per ogni $i \leq n$, il bit W_i può essere espresso come un polinomio binario di grado esattamente 2^i nelle variabili $\sigma_1, \ldots, \sigma_t$. Inoltre c'è un circuito aritmetico di dimensione $2^i \cdot t$ che simultaneamente calcola tutti i polinomi per $W_0, \ldots, W_i$.*

Dimostrazione. L'i-esimo bit nella rappresentazione binaria del peso di Hamming del vettore $\vec{\sigma}$ è uguale a $e_{2^i}(\vec{\sigma})$ modulo 2, dove $e_k(\cdot)$ è il k-esimo polinomio simmetrico elementare:

$$W_i = e_{2^i}(\vec{\sigma}) \bmod 2 = \Big(\sum_{|S|=2^i} \prod_{j \in S} \sigma_j \Big) \bmod 2,$$

dove il grado di e_{2^i} è esattamente 2^i.
Possiamo poi calcolare i polinomi simmetrici elementari nelle σ_i come coefficienti del polinomio $P_{\vec{\sigma}}(z) = \prod_{i=1}^{t}(z - \sigma_i)$ nella variabile ausiliaria z, in cui $e_k(\vec{\sigma})$ è il coefficiente di z^{t-k}. Per calcolare solo i primi bit $W_0, W_1, \ldots, W_i$ possiamo scartare i termini di grado più basso in $P_{\vec{\sigma}}(z)$, cioè non abbiamo bisogno dei coefficienti di z^j per $j < t - 2^i$.
Per esempio, una procedura per calcolare $W_0, W_1, \ldots, W_i$ può essere data dal seguente pseudocodice:

Input : i bit $\sigma_1, \ldots, \sigma_t$

0 . Inizializzazione: Si pone $P_{0,0} \leftarrow 1$ e $P_{j,0} \leftarrow 0$ per $j = 1, 2, 3, \ldots, 2^i$
// $P_{j,k}$ è il j-esimo polinomio simmetrico in $\sigma_1, \ldots, \sigma_k$

1 . Per $k = 1, 2, \ldots, t$ *// incorpora σ_k*

2 . Per $j = 2^i$ a 1, si pone $P_{j,k} \leftarrow \sigma_k \times P_{j-1,k-1} + P_{j,k-1}$

3 . **Output**: $P_{1,t}, P_{2,t}, P_{4,t}, \ldots, P_{2^i,t}$

Possiamo ottenere un miglioramento usando la trasformata di Fourier veloce, o FFT (dall'inglese fast Fourier trasform), per la moltiplicazione di polinomi. Usando questa tecnica possiamo calcolare l'intero polinomio $P_{\vec{\sigma}}(z)$ con complessità $t \cdot \text{poly}\log(t)$. □

1.3.3 Sicurezza del crittosistema modificato

Introducendo $\mathbf{y}$ nella chiave pubblica, abbiamo bisogno di un altro problema computazionalmente difficile, che garantisca comunque la sicurezza del crittosistema, nonostante siano state fornite delle informazioni aggiuntive nella chiave pubblica. Tale problema è lo Sparse Subset Sum Problem (SSSP), che consiste in una variante del più comune Subset Sum Problem, importante nella crittografia perché NP-completo. Il Subset Sum Problem è il seguente: dato un insieme $S \subset \mathbb{Z}$ e fissato un valore t, si ricerca un sottoinsieme di S, per cui la somma degli elementi che lo costituiscono sia uguale a t; nello Sparse Subset Sum Problem chiediamo inoltre che la cardinalità del sottoinsieme cercato sia molto minore della cardinalità di S. Lo SSSP è anche denominato "low-weight" knapsack problem perché costituisce una versione particolare del più noto problema dello zaino (Knapsack Problem): dato uno zaino che possa sopportare un determinato peso W e dati N oggetti, ognuno caratterizzato da un peso w_i e da un valore c_i, il problema si propone di scegliere quali di questi oggetti mettere nello zaino per ottenere il maggior valore senza eccedere nel peso sostenibile dallo zaino stesso. Più formalmente si vuole massimizzare $\sum_{i=1}^{N} c_i \cdot x_i$, dove x_i esprime la possibilità che un oggetto venga inserito o meno nello zaino, sapendo che il vincolo è $\sum_{i=1}^{N} w_i \cdot x_i \leq W$. Anche il problema dello zaino è NP-completo.

Rimandiamo all'Appendice B per le definizioni delle classi di complessità tra cui la classe NP-completo, facenti parte della teoria della complessità computazionale, e nel terzo capitolo vedremo come Gentry ha utilizzato lo SSSP nel contesto della crittografia pienamente omomorfa basata sui reticoli [12].

Tornando al caso attualmente trattato, possiamo scongiurare attacchi noti scegliendo θ grande abbastanza da evitare attacchi a forza bruta (a ricerca esaustiva) e ponendo Θ maggiore di $\omega(\log \lambda)$ volte il numero di bit dei numeri razionali nella chiave pubblica[2] [10].

[2]Il problema dell'approximate-gcd e lo Sparse Subset Sum Problem condividono lo stesso intero p, ma ciò non costituisce un ostacolo, in quanto SSSP è considerato difficile anche se si conosce p.

Capitolo 2

Reticoli

In questo capitolo provvederemo a fornire le nozioni necessarie alla comprensione della crittografia basata sui reticoli (*lattice-based cryptography*). I reticoli furono inizialmente studiati da Joseph Louis Lagrange (1736-1813) e Carl Friedrich Gauss (1777-1855) e rivestono un ruolo significativo nella crittografia odierna. Il loro primo uso in crittografia risale al 1996, grazie al lavoro di Miklós Ajtai [2], ma sono stati utilizzati in molti altri metodi; uno dei più famosi è NTRU (un metodo crittografico a chiave pubblica basato sui reticoli inventato da J. Hoffstein, J. Pipher e J.H. Silverman nel 1998 [19]).

Definizione 2.1. *Sia $\mathbb{R}^m$ lo spazio euclideo m-dimensionale, un reticolo in $\mathbb{R}^m$ è l'insieme*

$$\mathcal{L}(\mathbf{b}_1, ..., \mathbf{b}_n) = \Big\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \, : \, x_i \in \mathbb{Z} \Big\} \qquad (1)$$

di tutte le combinazioni a coefficienti interi di n vettori linearmente indipendenti $\mathbf{b}_1, ..., \mathbf{b}_n$ in $\mathbb{R}^m$ $(m \geq n)$.
Gli interi n e m sono chiamati il rango e la dimensione del reticolo rispettivamente.
I vettori $\mathbf{b}_1, ..., \mathbf{b}_n$ costituiscono una base per il reticolo e sono rappresentati da una matrice

$$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] \in M_{m,n}(\mathbb{R})$$

avente i vettori di base come colonne. Usando la notazione matriciale la (1) *può essere riscritta in una forma più compatta come segue*

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \, : \, \mathbf{x} \in \mathbb{Z}^n\} \qquad (2)$$

dove $\mathbf{Bx}$ è l'usuale moltiplicazione matrice-vettore.

Gli elementi del reticolo saranno chiamati vettori o punti. Graficamente un reticolo può essere descritto come l'insieme dei punti di intersezione di una griglia n-dimensionale, infinita, regolare (ma non necessariamente ortogonale). Un esempio 2-dimensionale è mostrato in Figura 2.1, nel quale i vettori di base sono

$$\mathbf{b}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \mathbf{b}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

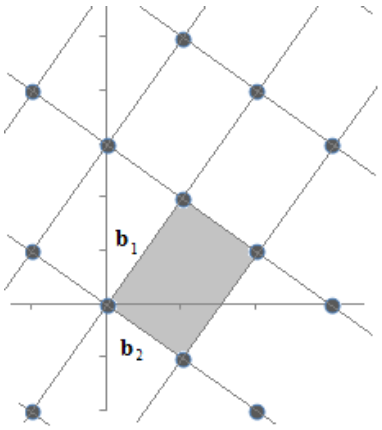

Figura 2.1: Un reticolo in $\mathbb{R}^2$

Lo stesso reticolo ha diverse basi: per esempio i vettori $\mathbf{b'}_1$ e $\mathbf{b'}_2$, definiti di seguito, costituiscono un'altra base per il reticolo $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$. La griglia generata da

$$\mathbf{b'}_1 = \mathbf{b}_1 + \mathbf{b}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad \mathbf{b'}_2 = 2\mathbf{b}_1 + \mathbf{b}_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

è mostrata in Figura 2.2.

Notiamo che, nonostante le due griglie siano differenti, l'insieme dei punti di intersezione è esattamente lo stesso, ovvero $\{\mathbf{b}_1, \mathbf{b}_2\}$ e $\{\mathbf{b'}_1, \mathbf{b'}_2\}$ sono due basi diverse per lo stesso reticolo; in altre parole: $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2) = \mathcal{L}(\mathbf{b'}_1, \mathbf{b'}_2)$. Infatti, preso un punto $(x, y)^T$ del reticolo $\mathcal{L}(\mathbf{b'}_1, \mathbf{b'}_2)$, esso si può scrivere come combinazione lineare a coefficienti interi ($a_1, a_2 \in \mathbb{Z}$) dei vettori di base $\mathbf{b'}_1, \mathbf{b'}_2$:

$$(x, y)^T = a_1\mathbf{b}'_1 + a_2\mathbf{b}'_2 = a_1(\mathbf{b}_1 + \mathbf{b}_2) + a_2(2\mathbf{b}_1 + \mathbf{b}_2) = (a_1 + 2a_2)\mathbf{b}_1 + (a_1 + a_2)\mathbf{b}_2,$$

che è una combinazione lineare sempre a coefficienti interi, ma dei vettori di base $\{\mathbf{b}_1, \mathbf{b}_2\}$, pertanto $(x, y)^T \in \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$. Viceversa, preso un punto $(x, y)^T \in \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$ e $c_1, c_2 \in \mathbb{Z}$, possiamo scrivere

$$(x, y)^T = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 = c_1(-\mathbf{b}'_1 + \mathbf{b}'_2) + c_2(2\mathbf{b}'_1 - \mathbf{b}'_2) = (-c_1 + 2c_2)\mathbf{b}'_1 + (c_1 - c_2)\mathbf{b}'_2$$

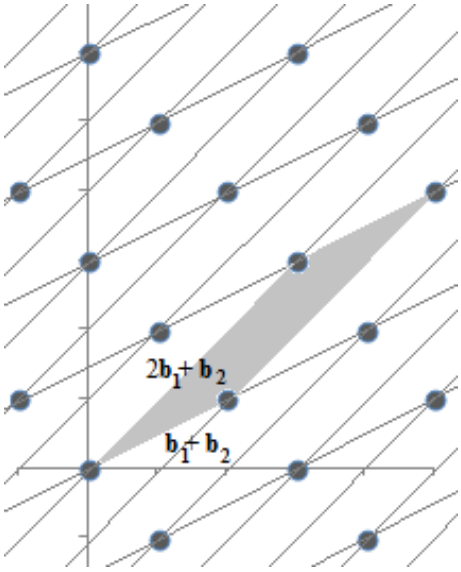

Figura 2.2: Una base diversa

e, quindi, $(x, y)^T \in \mathcal{L}(\mathbf{b}'_1, \mathbf{b}'_2)$.
Un semplice esempio di un reticolo n-dimensionale è dato dall'insieme $\mathbb{Z}^n$ di tutti i vettori con coordinate intere, di cui una possibile base è la base canonica

$$\mathbf{e}_i = [\overbrace{\underbrace{0, ..., 0, 1}_{i}, 0, ..., 0}^{n}]^T.$$

In notazione matriciale possiamo scrivere $\mathbb{Z}^n = \mathcal{L}(\mathbf{I})$ dove $\mathbf{I} \in M_{n,n}(\mathbb{Z})$ è la matrice identica n-dimensionale.
Quando $n = m$, i.e. il numero di vettori di base eguaglia il numero di coordinate, diciamo che $\mathcal{L}(\mathbf{B})$ ha *rango massimo* o dimensione massima; equivalentemente, il reticolo $\mathcal{L}(\mathbf{B}) \subseteq \mathbb{R}^m$ ha rango massimo se e solo se il sottospazio vettoriale generato dalla base $\mathbf{B}$

$$\text{span}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{R}^n\} \qquad (3)$$

coincide con l'intero spazio $\mathbb{R}^m$.
La differenza tra (2) e (3) sta nel fatto che in (3) si usano coefficienti reali, mentre in (2) sono permessi solo i coefficienti interi. Osserviamo inoltre che span($\mathbf{B}$) non dipende dalla particolare base, infatti se $\mathbf{B}$ e $\mathbf{B}'$ generano lo stesso reticolo allora span($\mathbf{B}$) = span($\mathbf{B}'$). In tal modo per ogni reticolo $\Lambda = \mathcal{L}(\mathbf{B})$ possiamo definire, con abuso di notazione, il sottospazio lineare span(Λ), senza riferirci ad una base particolare.
Notiamo che $\mathbf{B}$ è una base per span($\mathbf{B}$) come spazio vettoriale; in particolare, il rango del

reticolo $\mathcal{L}(\mathbf{B})$ è uguale alla dimensione di span$(\mathbf{B})$ come spazio vettoriale su $\mathbb{R}$ e costituisce un invariante per il reticolo, ovvero non dipende dalla scelta della base. Chiaramente, ogni insieme di n vettori linearmente indipendenti $\mathbf{B}' \in \mathcal{L}(\mathbf{B})$ è una base per span$(\mathbf{B})$ come spazio vettoriale, però non è necessariamente una base per il reticolo $\mathcal{L}(\mathbf{B})$, si veda la Figura 2.3 per un esempio 2-dimensionale. L'immagine mostra il reticolo $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$ generato dai vettori di base

$$\mathbf{b}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \mathbf{b}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

come in precedenza per la Figura 2.1, e la griglia associata ai vettori

$$\mathbf{b'}_1 = \mathbf{b}_1 + \mathbf{b}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad \mathbf{b'}_2 = \mathbf{b}_1 - \mathbf{b}_2 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}.$$

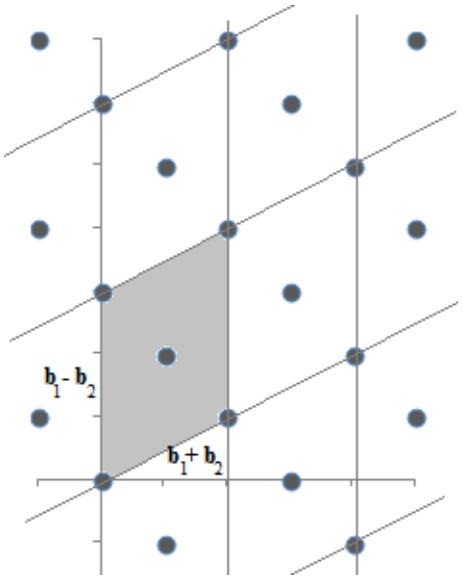

Figura 2.3: Il sottoreticolo generato da $\mathbf{b}_1 + \mathbf{b}_2$ e $\mathbf{b}_1 - \mathbf{b}_2$

I vettori $\mathbf{b'}_1$ e $\mathbf{b'}_2$ sono linearmente indipendenti e, quindi, sono una base per il piano $\mathbb{R}^2 =$ span$(\mathbf{b}_1, \mathbf{b}_2)$:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = a_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 3 \end{bmatrix} \Leftrightarrow \begin{cases} 0 &= 2a_1 \\ 0 &= a_1 + 3a_3 \end{cases} \Leftrightarrow \begin{cases} a_1 = 0 \\ a_3 = 0 \end{cases};$$

tuttavia essi non formano una base per $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$ perché il punto del reticolo relativo a $\mathbf{b}_1$ non può essere espresso come combinazione lineare intera di $\mathbf{b'}_1$ e $\mathbf{b'}_2$:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} = a_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 3 \end{bmatrix} \Leftrightarrow \begin{cases} 1 &= 2a_1 \\ 2 &= a_1 + 3a_3 \end{cases} \Leftrightarrow \begin{cases} a_1 = 1/2 \\ a_3 = 1/2 \end{cases}.$$

Diamo allora una caratterizzazione per i vettori del reticolo linearmente indipendenti che generano l'intero reticolo, partendo dalla seguente definizione:

Definizione 2.2. *Dato un reticolo Λ di dimensione n, generato dalla base $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, viene chiamato* ***dominio fondamentale*** *di Λ la porzione di spazio*

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} : 0 \leq x_i < 1\},$$

che, geometricamente, rappresenta il parallelepipedo "mezzo aperto" i cui spigoli sono i vettori della base.

Si ha allora che una famiglia di vettori linearmente indipendenti $\mathbf{V}$ è una base per il reticolo $\mathcal{L}(\mathbf{B})$ se e solo se $\mathcal{P}(\mathbf{V})$ non contiene alcun vettore del reticolo eccetto il vettore nullo.
Tornando ai nostri esempi, possiamo notare che nella Figura 2.2 il parallelepipedo mezzo aperto $\mathcal{P}(\mathbf{B}')$ non contiene alcun punto del reticolo eccetto il vettore nullo e, pertanto, possiamo affermare che $\mathcal{L}(\mathbf{B}') = \mathcal{L}(\mathbf{B})$; mentre nella Figura 2.3 osserviamo che $\mathcal{P}(\mathbf{B}')$ contiene il punto $\mathbf{b}_1$, quindi $\mathcal{L}(\mathbf{B}') \neq \mathcal{L}(\mathbf{B})$ e possiamo concludere che $\mathbf{B}'$ non è una base per $\mathcal{L}(\mathbf{B})$.
Dato un reticolo $\mathcal{L}(\mathbf{V}_1)$ generato dalla base $\mathbf{V}_1$, notiamo che, se $\mathbf{V}_2 \subseteq \mathcal{L}(\mathbf{V}_1)$ è un insieme di vettori linearmente indipendenti, $\mathcal{L}(\mathbf{V}_2)$ è un reticolo e $\mathbf{V}_2$ è una base per $\mathcal{L}(\mathbf{V}_2)$. Scriviamo $\mathcal{L}(\mathbf{V}_2) \subseteq \mathcal{L}(\mathbf{V}_1)$ per dire che ogni punto dal reticolo $\mathcal{L}(\mathbf{V}_2)$ appartiene anche a $\mathcal{L}(\mathbf{V}_1)$ e lo chiamiamo *sottoreticolo* di $\mathcal{L}(\mathbf{V_1})$. Se $\mathcal{L}(\mathbf{V}_2) = \mathcal{L}(\mathbf{V}_1)$ diciamo che le basi $\mathbf{V}_1$ e $\mathbf{V}_2$ sono *equivalenti*. Ricordiamo che basi equivalenti sono caratterizzate algebricamente come segue.

Osservazione 2.3. *Due basi $\mathbf{B}, \mathbf{B}' \in M_{m,n}(\mathbb{R})$ sono equivalenti se e solo se esiste una matrice $\mathbf{U} \in M_{n,n}(\mathbb{Z}), \det(\mathbf{U}) = \pm 1$ tale che $\mathbf{B} = \mathbf{B}'\mathbf{U}$.*

Dimostrazione. Innanzitutto se assumiamo che $\mathbf{B} = \mathbf{UB}'$ per qualche matrice unimodulare $\mathbf{U}$, allora si ha che anche l'inversa $\mathbf{U^{-1}}$ ha determinante uguale a ± 1 e $\mathbf{U^{-1}} \in M_{n,n}(\mathbb{Z})$; pertanto $\mathbf{B} = \mathbf{UB}'$ e $\mathbf{B}' = \mathbf{U}^{-1}\mathbf{B}$ da cui segue che $\mathcal{L}(\mathbf{B}) \subseteq \mathcal{L}(\mathbf{B}')$ e $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$, ovvero le due matrici $\mathbf{B}$ e $\mathbf{B}'$ generano lo stesso reticolo.
Viceversa, assumiamo che $\mathbf{B}$ e $\mathbf{B}'$ siano due basi per lo stesso reticolo; in tal caso esistono due matrici intere, $\mathbf{V}$ e $\mathbf{W}$, tali che $\mathbf{B} = \mathbf{B}'\mathbf{V}$ e $\mathbf{B}' = \mathbf{BW}$. Combinando queste due relazioni otteniamo $\mathbf{B} = \mathbf{BVW}$ o, equivalentemente, $\mathbf{B}(\mathbf{I} - \mathbf{VW}) = \mathbf{0}$. Dato che i vettori colonna di $\mathbf{B}$ sono linearmente indipendenti, si deve avere $\mathbf{I} - \mathbf{VW} = \mathbf{0}$, i.e. $\mathbf{VW} = \mathbf{I}$, da cui $\det(\mathbf{V}) \cdot \det(\mathbf{W}) = \det(\mathbf{V} \cdot \mathbf{W}) = \det(\mathbf{I}) = 1$, essendo $\mathbf{V}$ e $\mathbf{W}$ matrici intere, $\det(\mathbf{V})$, $\det(\mathbf{W}) \in \mathbb{Z}$ e pertanto deve essere $\det(\mathbf{V}) = \det(\mathbf{W}) = \pm 1$. □

Quando studiamo i reticoli da un punto di vista computazionale, i numeri reali sono approssimati a numeri razionali e, pertanto, considereremo i vettori di base (e quindi tutti i vettori

del reticolo) a coordinate razionali. È facile però notare che i reticoli razionali possono essere convertiti in reticoli interi (sottoreticoli di $\mathbb{Z}^n$) moltiplicando tutte le coordinate per un fattore intero appropriato. Da ora in poi considereremo pertanto solo reticoli interi e assumeremo che siano rappresentati da una base, ossia una matrice intera con le colonne linearmente indipendenti.
Tuttavia i reticoli possono essere anche caratterizzati senza riferirsi a una base; in tal caso serve però la definizione di norma e distanza, che riportiamo qui di seguito.
Ricordiamo che la norma è una funzione omogenea, definita positiva, che soddisfa la disuguaglianza triangolare, i.e. una funzione $||\cdot|| : \mathbb{R}^n \to \mathbb{R}$ tale che

- $||\mathbf{x}|| \geq 0$ (con $||\mathbf{x}|| = 0$ solo se $\mathbf{x} = 0$),
- $||\alpha\mathbf{x}|| = |\alpha| \cdot ||\mathbf{x}||$,
- $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$,

per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ e $\alpha \in \mathbb{R}$. Un'importante famiglia di funzioni norma è data dalla norma l_p: per ogni $p \geq 1$, la norma l_p di un vettore $\mathbf{x} \in \mathbb{R}^n$ è

$$||\mathbf{x}||_p = \Big(\sum_{i=1}^{n} x_i^p \Big)^{1/p}.$$

Riportiamo alcuni importanti casi particolari:

- la norma l_1, $||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$,
- la norma l_2 (o norma Euclidea), $||\mathbf{x}||_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$,
- la norma l_∞, $||\mathbf{x}||_\infty = \lim_{p\to\infty} ||\mathbf{x}||_p = \max_{i=1}^{n} |x_i|$.

Una distanza (o metrica) su un insieme X è una funzione $d : X \times X \to \mathbb{R}$ che soddisfa le seguenti proprietà per ogni $x, y, z \in X$:

- $d(x, y) \geq 0$,
- $d(x, y) = 0$ se e solo se $x = y$,
- $d(x, y) = d(y, x)$,
- $d(x, y) \leq d(x, z) + d(z, y)$ (disuguaglianza triangolare).

Data una norma $||\cdot|| : X \to \mathbb{R}$, con X uno spazio vettoriale reale o complesso, è possibile definire una distanza $d : X \times X \leftarrow \mathbb{R}$, ponendo $d(x, y) = ||x - y||$. Si verifica infatti che la definizione data soddisfa le proprietà sopra elencate:

- $d(x, y) = ||x - y|| \geq 0$,
- $d(x, y) = ||x - y|| = 0$ se e solo se $x = y$,
- $d(x, y) = ||x - y|| = ||-1(y - x)|| = |-1| \cdot ||y - x|| = d(y, x)$,
- $d(x, y) = ||x - y|| = ||x - z + z - y|| \leq ||x - z|| + ||z - y|| = d(x, z) + d(z, y)$.

Noi ci concentreremo principalmente sulla norma l_2, che corrisponde alla familiare distanza Euclidea:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Nel seguito, se non sarà specificato diversamente, lavoreremo con la norma euclidea e pertanto ometteremo l'indice usualmente presente nel simbolo di norma.
Diamo ora una definizione alternativa di reticolo.

Definizione 2.4. *Un reticolo Λ può essere definito come un sottogruppo additivo discreto non vuoto di $\mathbb{R}^m$, i.e. un sottoinsieme $\Lambda \subseteq \mathbb{R}^m$ che soddisfa le seguenti proprietà:*
(sottogruppo) Λ *è chiuso rispetto alla sottrazione, ovvero se* $\mathbf{x}, \mathbf{y} \in \Lambda$ *allora* $\mathbf{x} - \mathbf{y} \in \Lambda$,
(discreto) $\exists \lambda > 0$ *in* $\mathbb{R}$ *tale che,* $\forall \mathbf{x} \neq \mathbf{y}$ *vettori di* Λ, *la distanza tra* $\mathbf{x}$ *e* $\mathbf{y}$ *è almeno* λ.

Un tipico esempio è l'insieme $\Lambda = \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{Ax} = \mathbf{0}\}$ delle soluzioni intere di un sistema di equazioni lineari omogenee: se $\mathbf{x}, \mathbf{y} \in \Lambda$ allora $\mathbf{A}(\mathbf{x} - \mathbf{y}) = \mathbf{Ax} - \mathbf{Ay} = \mathbf{0}$, risultando chiuso rispetto alla sottrazione, ed è discreto, in quanto $\forall \mathbf{x} \neq \mathbf{y}$ vettori di Λ, la distanza tra $\mathbf{x}$ e $\mathbf{y}$ è $||\mathbf{x} - \mathbf{y}|| > \lambda$, prendendo $\lambda = 1/2$; in altre parole, Λ è un sottogruppo additivo discreto di $\mathbb{R}^m$.

2.1 Il determinante

Il *determinante* di un reticolo $\Lambda = \mathcal{L}(\mathbf{B})$, denotato con $\det(\Lambda)$, è il volume n-dimensionale del parallelepipedo fondamentale $\mathcal{P}(\mathbf{B})$ generato dai vettori di base.
Il determinante è un invariante del reticolo, ovvero non dipende dalla particolare base usata per calcolarlo. Ciò segue immediatamente dalla caratterizzazione delle basi equivalenti come matrici $\mathbf{B}' = \mathbf{BU}$ relative ad una trasformazione unimodulare $\mathbf{U}$.

Un modo per calcolare il determinante si ha tramite il processo di *ortogonalizzazione di Gram-Schmidt*; per ogni n-upla di vettori $\mathbf{b}_1, ..., \mathbf{b}_n$ definiamo i corrispondenti vettori ortogonalizzati $\mathbf{b}_1^*, ..., \mathbf{b}_n^*$ come segue:

$$\mathbf{b_i^*} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b_j^*} \qquad \text{dove} \qquad \mu_{i,j} = \frac{\langle \mathbf{b_i}, \mathbf{b_j^*} \rangle}{\langle \mathbf{b_j^*}, \mathbf{b_j^*} \rangle},$$

con l'usuale notazione, $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{m} x_i y_i$. Si ottengono vettori a due a due ortogonali, i.e. $\langle \mathbf{b_i^*}, \mathbf{b_j^*} \rangle = 0$; inoltre, $\forall i$, $\text{span}\{\mathbf{b}_1, \ldots, \mathbf{b}_i\} = \text{span}\{\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*\}$.
Il determinante del reticolo è uguale al prodotto delle lunghezze dei vettori ortogonalizzati

$$\det(\mathcal{L}(\mathbf{B})) = \text{vol}(\mathcal{P}(\mathbf{B})) = \prod_{i=1}^{n} ||\mathbf{b_i^*}||,$$

dove $||\mathbf{x}|| = \sqrt{\sum_i x_i^2}$ è l'usuale lunghezza Euclidea. Intuitivamente, più il determinante aumenta più i punti sono distanti in $\mathcal{L}(\mathbf{B})$.

Proposizione 2.5. *Per ogni base* $\mathbf{B}$*, rappresentata da una matrice in* $M_{m,n}(\mathbb{R})$*, di un reticolo* $\mathcal{L}(\mathbf{B})$

$$\det(\mathcal{L}(\mathbf{B})) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}.$$

In particolare, se $\mathbf{B} \in M_{m,n}(\mathbb{R})$ *è una matrice quadrata (non singolare) allora* $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$.

Dimostrazione. Ricordando la procedura di ortogonalizzazione di Gram-Schmidt, possiamo scrivere in notazione matriciale i vettori ortogonalizzati $\mathbf{B}^*$ secondo la relazione $\mathbf{B} = \mathbf{B}^*\mathbf{T}$, dove $\mathbf{T}$ è una matrice triangolare superiore con tutti 1 sulla diagonale e i coefficienti $\mu_{i,j}$ nella posizione (j, i) per ogni $j < i$. Si ha allora

$$\sqrt{\det(\mathbf{B}^T\mathbf{B})} = \sqrt{\det(\mathbf{T}^T\mathbf{B}^{*T}\mathbf{B}^*\mathbf{T})} = \sqrt{\det(\mathbf{T}^T)\ \det(\mathbf{B}^{*T}\mathbf{B}^*)\ \det(\mathbf{T})}.$$

Le matrici $\mathbf{T}^T$ e $\mathbf{T}$ sono triangolari e i loro determinanti possono essere facilmente calcolati come il prodotto degli elementi in diagonale, che risulta uguale a 1. Consideriamo poi $\mathbf{B}^{*T}\mathbf{B}^*$, è una matrice diagonale perché le colonne di $\mathbf{B}^*$ sono ortogonali e tutti gli elementi fuori dalla diagonale risultano nulli. Pertanto

$$\det(\mathbf{B}^{*T}\mathbf{B}^*) = \prod_i \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle = (\prod_i ||\mathbf{b}_i^*||)^2 = \det(\mathcal{L}(\mathbf{B}))^2$$

e, passando alla radice quadrata, si ottiene

$$\sqrt{\det(\mathbf{T}^T)\ \det(\mathbf{B}^{*T}\mathbf{B}^*)\ \det(\mathbf{T})} = \det(\mathcal{L}(\mathbf{B})). \qquad \square$$

La proposizione mostra una via alternativa per calcolare il determinante di un reticolo, rispetto all'ortogonalizzazione dei vettori tramite Gram-Schmidt.

Data una base qualsiasi $\mathbf{B}$ di $\mathcal{L}$, si può calcolare efficientemente la sua forma normale di Hermite HNF($\mathcal{L}$). Con l'avverbio "efficientemente" intendiamo in un tempo polinomiale nella dimensione del reticolo e nella lunghezza del vettore colonna più lungo di $\mathbf{B}$. HNF($\mathcal{L}$) non rivela maggiori informazioni sulla struttura del reticolo rispetto alle altre basi, risultando un buon candidato per la chiave pubblica.

Definizione 2.6. *Una matrice non-singolare, quadrata $\mathbf{B} = [b_{i,j}] \in M_{n,n}(\mathbb{R})$ è nella* ***forma normale Hermitiana*** *(HNF) se e solo se:*

- $\mathbf{B}$ *è una matrice triangolare superiore ($b_{i,j} \neq 0 \Rightarrow i \leq j$),*
- *ogni riga di* $\mathbf{B}$ *ha un unico valore massimo che si trova sulla diagonale principale (per ogni $i < j$, $b_{i,j} < b_{i,i}$).*

Precisiamo che viene considerata in forma normale hermitiana anche una matrice quadrata triangolare inferiore. Ora generalizziamo la precedente definizione a matrici non quadrate, limitandoci a considerare il caso di nostro interesse.

Definizione 2.7. *Diciamo che una matrice $\mathbf{B} = [b_{i,j}] \in M_{m,n}(\mathbb{R})$, con $m \geq n$ e di rango n, è nella forma normale di Hermite (HNF) se esiste una sottomatrice quadrata $n \times n$ che soddisfa la Definizione 2.6 precedente, ossia se esistono n righe tali che, presa tra di esse la k-esima riga, il suo k-esimo elemento è un numero con al di sotto in colonna solo zeri e alla sua destra in riga solo elementi più piccoli.*

Esempio. Consideriamo la matrice 5×2 nella Figura 2.4: essa è in forma normale hermitiana.

Definizione 2.8. *Il reticolo duale di $\mathcal{L}$, denotato con $\mathcal{L}^*$, è definito come*

$$\mathcal{L}^* = \{\mathbf{x} \in \text{span}(\mathcal{L}) : \forall \mathbf{v} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\}.$$

Si ha $\det(\mathcal{L}) \cdot \det(\mathcal{L}^*) = 1$ e, se il reticolo ha rango massimo e $\mathbf{B}$ è una sua base, allora $(\mathbf{B}^{-1})^T$ (la trasposta dell'inversa di $\mathbf{B}$) è una base per $\mathcal{L}^*$. Formalizzando quanto detto, abbiamo il seguente teorema.

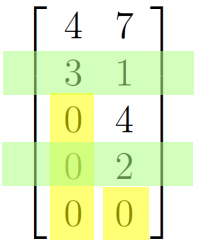

Figura 2.4: Un esempio di matrice 5×2, in cui è evidenziata in verde la sottomatrice quadrata 2×2 che risponde alla Definizione 2.6, in giallo è stato messo in risalto che, presa la prima riga verde (3 1), il suo primo elemento, 3, ha al di sotto in colonna solo zeri e a destra nella stessa riga elementi minori, $1 < 3$, e, presa la seconda riga verde (0 2), il suo secondo elemento, 2, ha al di sotto in colonna solo zeri.

Teorema 2.9. *Dato un reticolo $\mathcal{L}(\mathbf{B})$ di rango massimo, sia $\mathbf{B}$ una sua base, allora $(\mathbf{B}^{-1})^T$ è una base per il reticolo duale $\mathcal{L}^*$.*

Dimostrazione. Denotiamo con $\mathcal{L}'$ il reticolo generato dalla trasposta dell'inversa di $\mathbf{B}$: $\mathcal{L}' = \mathcal{L}((\mathbf{B}^{-1})^T)$. Dimostriamo la doppia inclusione:

• $\mathcal{L}' \subseteq \mathcal{L}^*$: per ogni $\mathbf{y}, \mathbf{v} \in \mathbb{Z}^n$, l'elemento di $\mathcal{L}'$ è $(\mathbf{B}^{-1})^T\mathbf{y}$ e si ha

$$\langle (\mathbf{B}^{-1})^T\mathbf{y}, \mathbf{B}\mathbf{v} \rangle = ((\mathbf{B}^{-1})^T\mathbf{y})^T\mathbf{B}\mathbf{v} = \mathbf{y}^T\mathbf{B}^{-1}\mathbf{B}\mathbf{v} = \mathbf{y}^T\mathbf{v} \in \mathbb{Z}$$

e quindi $\mathbf{y} \in \mathcal{L}^*$.

• $\mathcal{L}^* \subseteq \mathcal{L}'$: per ogni $\mathbf{y} \in \mathcal{L}^*$, poniamo $\mathbf{s}^T = \mathbf{y}^T\mathbf{B}$ così, per definizione, $\mathbf{s} \in \mathbb{Z}^n$. Da ciò abbiamo che $\mathbf{y}^T = \mathbf{s}^T\mathbf{B}^{-1}$, quindi $\mathbf{y} = (\mathbf{B}^{-1})^T\mathbf{s} \in \mathcal{L}'$. □

Definizione 2.10. *Per un vettore $\mathbf{v} \in \mathbb{R}^m$, indichiamo con $\mathbf{v} \bmod \mathbf{B}$ l'unico vettore $\widetilde{\mathbf{v}} \in \mathcal{P}(\mathbf{B})$ tale che $\mathbf{v} - \widetilde{\mathbf{v}} \in \mathcal{L}$.*

Dati $\mathbf{v}$ e $\mathbf{B}$, $\mathbf{v} \bmod \mathbf{B}$ può essere calcolato nel modo seguente

$$\mathbf{v} \bmod \mathbf{B} = \mathbf{v} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{v} \rceil = \mathbf{B} \cdot \big(\mathbf{B}^{-1} \cdot \mathbf{v} - \lfloor \mathbf{B}^{-1} \cdot \mathbf{v} \rceil\big), \tag{2.1}$$

dove $\lfloor \cdot \rceil$ approssima i coefficienti di un vettore agli interi più vicini; ad esempio, se $\mathbf{v} = [13/5, 18/7, 7/3]^T$, risulta $\lfloor \mathbf{v} \rceil = [3, 3, 2]^T$ e $\mathbf{v} - \lfloor \mathbf{v} \rceil = [-2/5, -3/7, 1/3]^T$. Osserviamo, infine, che per ogni base $\mathbf{B}$ si ha $||\mathbf{v} \bmod \mathbf{B}|| \geq \text{dist}(\mathcal{L}, \mathbf{v})$.

2.2 Minimi successivi

Sia $\mathcal{B}_m(\mathbf{0}, r) = \{\mathbf{x} \in \mathbb{R}^m \ : \ ||\mathbf{x}|| < r\}$ la palla aperta m-dimensionale di raggio r centrata in $\mathbf{0}$. Quando la dimensione m risulterà chiara dal contesto ometteremo il pedice m e scriveremo

semplicemente $\mathcal{B}(\mathbf{0}, r)$. Le costanti fondamentali associate ad ogni reticolo Λ di rango n sono i suoi minimi successivi $\lambda_1, ..., \lambda_n$, dove l'$i$-esimo minimo $\lambda_i(\Lambda)$ è il raggio della più piccola sfera centrata nell'origine contenente i vettori linearmente indipendenti del reticolo

$$\lambda_i(\Lambda) = \inf\{r\colon \dim(\mathrm{span}(\Lambda \cap \mathcal{B}(\mathbf{0}, r))) \geq i\}.$$

I minimi successivi possono essere definiti rispetto ad ogni norma e notiamo che il valore dei minimi successivi $\lambda_1, ..., \lambda_n$ dipende dalla norma usata; consideriamo per esempio il reticolo

$$\Lambda = \{\mathbf{v} \in \mathbb{Z}^2 : v_1 + v_2 = 0 \bmod 2\}$$

generato dai vettori di base

$$\mathbf{b}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \qquad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Il vettore $\mathbf{b}_1$ è un vettore non nullo di lunghezza minima in $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$ rispetto alla norma l_1, dato che, per $\mathbf{0} \neq \mathbf{v} \in \Lambda$, $||\mathbf{v}||_1 = |v_1|+|v_2| > 0$, ma $|v_1|+|v_2| \neq 1$ dalla definizione di Λ, quindi $||\mathbf{v}||_1 \geq 2$ e $\lambda_1 = ||\mathbf{b}_1||_1 = 2$; però $\mathbf{b}_1$ non continua ad essere il vettore più corto rispetto alla norma l_2 o l_∞, perché in queste norme $\mathbf{b}_2$ è un vettore strettamente più corto di $\mathbf{b}_1$ ($\lambda_1 = ||\mathbf{b}_2||_2 = \sqrt{2}$ e $\lambda_1 = ||\mathbf{b}_2||_\infty = 1$, rispettivamente). Nell'esempio precedente abbiamo visto che il reticolo contiene un vettore $\mathbf{b}$ tale che $||\mathbf{b}|| = \lambda_1$ e ciò è vero per ogni reticolo data la caratterizzazione dei reticoli come sottogruppi discreti di $\mathbb{R}^n$. Di seguito mostreremo che ogni reticolo contiene vettori non nulli di lunghezza minima, nel fare ciò troveremo un limite inferiore per il primo minimo che sarà utile in un secondo momento. Il risultato è facilmente generalizzato a tutti i minimi successivi per mostrare che ci sono n vettori indipendenti $\mathbf{v}_1, \ldots, \mathbf{v}_n$ soddisfacenti $||\mathbf{v}_i|| = \lambda_i$ per ogni $i = 1, \ldots, n$. Fissiamo un reticolo $\mathcal{L}(\mathbf{B})$ e consideriamo

$$\lambda_1 = \inf\{||\mathbf{v}|| : \mathbf{v} \in \mathcal{L}(\mathbf{B})\backslash\{\mathbf{0}\}\}.$$

Vogliamo provare che esiste un vettore $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ tale che $||\mathbf{v}|| = \lambda_1$, dimostrando innanzitutto che λ_1 è strettamente positivo.

Teorema 2.11. *Sia* $\mathbf{B}$ *una base per un reticolo* Λ *e sia* $\mathbf{B}^*$ *la corrispondente ortogonalizzazione di Gram-Schmidt. Allora* λ_1 *del reticolo soddisfa (secondo la norma* l_2*)*

$$\lambda_1 \geq \min_j ||\mathbf{b}_j^*|| > 0.$$

Dimostrazione. Consideriamo un generico vettore non nullo $\mathbf{Bx}$ (dove $\mathbf{x} \in \mathbb{Z}^n$ e $\mathbf{x} \neq \mathbf{0}$) e sia i l'indice maggiore tale che $x_i \neq 0$. Mostriamo che $||\mathbf{Bx}|| \geq ||\mathbf{b}_i^*|| \geq \min_j ||\mathbf{b}_j^*||$; segue che l'estremo inferiore $\lambda_1 = \inf ||\mathbf{Bx}||$ soddisfa $\lambda_1 \geq \min_j ||\mathbf{b}_j^*||$. Dall'algebra lineare sappiamo $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq ||\mathbf{x}|| \cdot ||\mathbf{y}||$ per ogni coppia di vettori $\mathbf{x}, \mathbf{y}$. Proviamo che $|\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle| \geq ||\mathbf{b}_i^*||^2$ e quindi $||\mathbf{Bx}|| \cdot ||\mathbf{b}_i^*|| \geq ||\mathbf{b}_i^*||^2$; essendo $||\mathbf{b}_i^*|| \neq 0$ e segue $||\mathbf{Bx}|| \geq ||\mathbf{b}_i^*||$. Perciò il nostro obiettivo ora consiste nel provare che $|\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle| \geq ||\mathbf{b}_i^*||^2$. Dalla definizione di i sappiamo che $\mathbf{Bx} = \sum_{j=1}^{i} \mathbf{b}_j x_j$ e, usando la definizione di vettori ortogonalizzati, otteniamo

$$\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle = \sum_{j=1}^{i} \langle \mathbf{b}_j, \mathbf{b}_i^* \rangle x_j = \langle \mathbf{b}_i, \mathbf{b}_i^* \rangle x_i = \langle \mathbf{b}_i^* + \sum_{j<i} \mu_{ij} \mathbf{b}_j^*, \mathbf{b}_i^* \rangle x_i =$$

$$= \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle x_i + \sum_{j<i} \mu_{ij} \langle \mathbf{b}_j^*, \mathbf{b}_i^* \rangle x_i = ||\mathbf{b}_i^*||^2 x_i.$$

Dato che x_i è un intero non nullo,

$$|\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle| = ||\mathbf{b}_i^*||^2 |x_i| \geq ||\mathbf{b}_i^*||^2. \qquad \square$$

In particolare il teorema mostra che $\lambda_1 > 0$. Proviamo ora che esiste un vettore non nullo di lunghezza λ_1. Dalla definizione di λ_1 esiste una successione di vettori $\mathbf{v}_i \in \mathcal{L}(\mathbf{B})$ tali che

$$\lim_{i \to \infty} ||\mathbf{v}_i|| = \lambda_1.$$

Dato che $\lambda_1 > 0$, per tutti gli i sufficientemente grandi si deve avere $||\mathbf{v}_i|| \leq 2\lambda_1$, i.e. il vettore $\mathbf{v}_i$ appartiene alla palla chiusa $\overline{\mathcal{B}}(\mathbf{0}, 2\lambda_1) = \{\mathbf{z} : ||\mathbf{z}|| \leq 2\lambda_1\}$. $\overline{\mathcal{B}}(\mathbf{0}, 2\lambda)$ è compatto, possiamo estrarre una sottosequenza convergente $\mathbf{v}_{i_j}$ con limite

$$\mathbf{w} = \lim_{j \to \infty} \mathbf{v}_{i_j}.$$

Chiaramente $||\mathbf{w}|| = \lim_{j \to \infty} ||\mathbf{v}_{i_j}|| = \lambda_1$; vogliamo provare che $\mathbf{w}$ è un vettore del reticolo.
Per definizione di $\mathbf{w}$ abbiamo $\lim_{j \to \infty} ||\mathbf{v}_{i_j} - \mathbf{w}|| = \mathbf{0}$. Quindi per ogni j sufficientemente grande $||\mathbf{v}_{i_j} - \mathbf{w}|| < \lambda_1/2$. Dalla disuguaglianza triangolare per un j sufficientemente grande e per tutti i $k > j$,

$$||\mathbf{v}_{i_j} - \mathbf{v}_{i_k}|| \leq ||\mathbf{v}_{i_j} - \mathbf{w}|| + ||\mathbf{w} - \mathbf{v}_{i_k}|| < \lambda_1.$$

Ma $\mathbf{v}_{i_j} - \mathbf{v}_{i_k}$ è un vettore del reticolo e nessun vettore del reticolo non nullo può avere lunghezza strettamente minore di λ_1. Questo prova che $\mathbf{v}_{i_j} - \mathbf{v}_{i_k} = \mathbf{0}$, i. e. $\mathbf{v}_{i_k} = \mathbf{v}_{i_j}$ per tutti i $k > j$. Quindi $\mathbf{w} = \lim_k \mathbf{v}_{i_k} = \mathbf{v}_{i_j}$ e $\mathbf{w}$ è un vettore del reticolo. L'argomentazione appena vista può essere generalizzata per provare il seguente teorema su tutti i minimi successivi di un reticolo.

Teorema 2.12. *Sia Λ un reticolo di rango n con minimi successivi $\lambda_1, ..., \lambda_n$. Allora esistono n vettori linearmente indipendenti del reticolo $\mathbf{v}_1, ..., \mathbf{v}_n \in \Lambda$ tali che $||\mathbf{v}_i|| = \lambda_i$, $\forall i = 1, ..., n$.*

In particolare λ_1 è la lunghezza del vettore del reticolo non nullo più corto ed eguaglia la distanza minima tra due punti del reticolo

$$\lambda_1(\Lambda) = \min_{\mathbf{x} \neq \mathbf{y} \in \Lambda} ||\mathbf{x} - \mathbf{y}|| = \min_{\mathbf{x} \in \Lambda \setminus \{\mathbf{0}\}} ||\mathbf{x}||.$$

2.3 I teoremi di Minkowski

In questa sezione proveremo un importante limite superiore per il prodotto di tutti i minimi successivi di un reticolo. La dimostrazione di tale maggiorazione è basata sul teorema seguente.

Teorema 2.13 (**Blichfeldt**). *Per ogni reticolo Λ e per ogni insieme misurabile $S \subseteq \text{span}(\Lambda)$, se S ha volume $\text{vol}(S) > \det(\Lambda)$, allora esistono due punti distinti $\mathbf{z}_1, \mathbf{z}_2 \in S$ tale che $\mathbf{z}_1 - \mathbf{z}_2 \in \Lambda$.*

Dimostrazione. Sia $\Lambda = \mathcal{L}(\mathbf{B})$ un reticolo e S un sottoinsieme di $\text{span}(\Lambda)$ tale che $\text{vol}(S) > \det(\Lambda)$. Partizioniamo S in una collezione di regioni disgiunte come segue: per ogni vettore del reticolo $\mathbf{x} \in \Lambda$ definiamo

$$S_{\mathbf{x}} = S \cap (\mathcal{P}(\mathbf{B}) + \mathbf{x})$$

dove $\mathcal{P}(\mathbf{B})$ è il parallelepipedo mezzo aperto. Notiamo che gli insiemi $\mathcal{P}(\mathbf{B}) + \mathbf{x}$ (con $\mathbf{x} \in \Lambda$) formano una partizione per $\text{span}(\mathbf{B})$. Quindi gli insiemi $S_{\mathbf{x}}$ ($\mathbf{x} \in \Lambda$) formano una partizione per S, i.e. sono a due a due disgiunti e $S = \bigcup_{\mathbf{x} \in \Lambda} S_{\mathbf{x}}$. In particolare, dato che Λ è numerabile,

$$\text{vol}(S) = \sum_{\mathbf{x} \in \Lambda} \text{vol}(S_{\mathbf{x}}).$$

Definiamo inoltre gli insiemi traslati $S'_{\mathbf{x}} = S_{\mathbf{x}} - \mathbf{x} = (S - \mathbf{x}) \cap \mathcal{P}(\mathbf{B})$ e notiamo che, per ogni $\mathbf{x} \in \Lambda$, $S'_{\mathbf{x}}$ è contenuto in $\mathcal{P}(\mathbf{B})$ e $\text{vol}(S_{\mathbf{x}}) = \text{vol}(S'_{\mathbf{x}})$. Vogliamo ora dimostrare che gli insiemi $S'_{\mathbf{x}}$ non sono a due a due disgiunti; assumiamo, per assurdo, che lo siano. Allora abbiamo

$$\sum_{\mathbf{x} \in \Lambda} \text{vol}(S'_{\mathbf{x}}) = \text{vol}\Big(\bigcup_{\mathbf{x} \in \Lambda} S'_{\mathbf{x}}\Big) \leq \text{vol}(\mathcal{P}(\mathbf{B})).$$

Sappiamo anche dalle ipotesi del teorema

$$\sum_{\mathbf{x} \in \Lambda} \text{vol}(S'_{\mathbf{x}}) = \sum_{\mathbf{x} \in \Lambda} \text{vol}(S_{\mathbf{x}}) = \text{vol}(S) > \det(\Lambda).$$

Combinando le ultime due asserzioni, otteniamo la disuguaglianza $\det(\Lambda) < \text{vol}(\mathcal{P}(\mathbf{B}))$, in contraddizione con $\det(\Lambda) = \text{vol}(\mathcal{P}(\mathbf{B}))$ che segue dalla definizione del determinante di un reticolo. Questo prova che gli insiemi $S'_{\mathbf{x}}$ non sono a due a due disgiunti, ossia esistono due insiemi $S'_{\mathbf{x}}$ e $S'_{\mathbf{y}}$ per $\mathbf{x} \neq \mathbf{y} \in \Lambda$ tali che $S'_{\mathbf{x}} \cap S'_{\mathbf{y}} \neq \emptyset$. Sia $\mathbf{z}$ un vettore appartenente all'intersezione $S'_{\mathbf{x}} \cap S'_{\mathbf{y}}$ e definiamo

$$\mathbf{z}_1 = \mathbf{z} + \mathbf{x} \qquad \text{e} \qquad \mathbf{z}_2 = \mathbf{z} + \mathbf{y}.$$

Dato che $\mathbf{z} \in S'_{\mathbf{x}}$ e $\mathbf{z} \in S'_{\mathbf{y}}$, otteniamo $\mathbf{z}_1 \in S_{\mathbf{x}} \subseteq S$ e $\mathbf{z}_2 \in S_{\mathbf{y}} \subseteq S$; inoltre $\mathbf{z}_1 \neq \mathbf{z}_2$ perché $S_{\mathbf{x}} \cap S_{\mathbf{y}} = \emptyset$ dato che $\mathbf{x} \neq \mathbf{y}$. Infine la differenza tra $\mathbf{z}_1$ e $\mathbf{z}_2$ soddisfa

$$\mathbf{z}_1 - \mathbf{z}_2 = \mathbf{x} - \mathbf{y} \in \Lambda,$$

completando la dimostrazione. □

Come corollario del teorema di Blichfeldt segue immediatamente il seguente teorema di Minkowski.

Teorema 2.14 (**Il teorema del corpo convesso**). *Per ogni reticolo Λ di rango n e per ogni insieme convesso $S \subset span(\Lambda)$ simmetrico rispetto all'origine, se vol$(S) > 2^n \det(\Lambda)$ allora S contiene un punto del reticolo $\mathbf{v} \neq \mathbf{0}$.*

Dimostrazione. Consideriamo l'insieme $S' = \{\mathbf{x} : 2\mathbf{x} \in S\}$; il volume di S' soddisfa

$$\text{vol}(S') = 2^{-n}\text{vol}(S) > \det(\Lambda).$$

Per il teorema di Blichfeldt esistono due punti distinti $\mathbf{z}_1, \mathbf{z}_2 \in S'$ tali che $\mathbf{z}_1 - \mathbf{z}_2 \in \Lambda$. Dalla definizione di S' sappiamo che $2\mathbf{z}_1, 2\mathbf{z}_2 \in S$ e dato che S è simmetrico rispetto all'origine anche $-2\mathbf{z}_2 \in S$. Infine, per la convessità, il punto medio del segmento $[2\mathbf{z}_1, -2\mathbf{z}_2]$ appartiene a S, i.e.

$$\frac{2\mathbf{z}_1 + (-2\mathbf{z}_2)}{2} = \mathbf{z}_1 - \mathbf{z}_2 \in S.$$

Questo prova che $\mathbf{v} = \mathbf{z}_1 - \mathbf{z}_2$ è un punto non nullo del reticolo in S. □

Il teorema del corpo convesso di Minkowski può essere usato per limitare la lunghezza del vettore non nullo più corto in un reticolo di rango n come segue. Sia $S = \mathcal{B}(\mathbf{0}, \sqrt{n}\det(\Lambda)^{1/n}) \cap \text{span}(\Lambda)$ la palla aperta di raggio $\sqrt{n}\det(\Lambda)^{1/n}$ in span(Λ). Notiamo che S ha un volume strettamente più grande di $2^n\det(\Lambda)$ perché contiene un ipercubo n-dimensionale con spigoli di lunghezza $2\det(\Lambda)^{1/n}$. Dal teorema di Minkowski esiste un vettore del reticolo non nullo

tale che $\mathbf{v} \in S$, i.e. $||\mathbf{v}|| < \sqrt{n}\det(\Lambda)^{1/n}$. Questo prova che, per ogni reticolo Λ di rango n, la lunghezza del vettore non nullo più corto (rispetto alla norma l_2) soddisfa

$$\lambda_1 < \sqrt{n}\det(\Lambda)^{1/n}.$$

Questo risultato è conosciuto come il *primo teorema di Minkowski*. Il *secondo teorema di Minkowski* riguarda tutti i minimi successivi e afferma che $\sqrt{n}\det(\Lambda)^{1/n}$ è un limite superiore, non solo per il primo minimo, ma per tutti i minimi successivi. Precisiamo che, se il primo teorema è facilmente generalizzabile a qualsiasi norma, il secondo è più complesso e lo proveremo solo nel caso della norma Euclidea.

Teorema 2.15 (**Il secondo teorema di Minkowski**). *Per ogni reticolo $\mathcal{L}(\mathbf{B})$ di rango n, i minimi successivi (rispetto alla norma l_2) $\lambda_1, \ldots, \lambda_n$ soddisfano*

$$\Big(\prod_{i=1}^{n} \lambda_i\Big)^{1/n} < \sqrt{n}\det(\mathcal{L}(\mathbf{B}))^{1/n}.$$

Dimostrazione. Siano $\mathbf{x}_1, \ldots, \mathbf{x}_n$ vettori linearmente indipendenti tali che $||\mathbf{x}_i|| = \lambda_i$ e assumiamo per assurdo che $\Big(\prod_{i=1}^{n} \lambda_i\Big)^{1/n} \geq \sqrt{n}\ \det(\mathcal{L}(\mathbf{B}))^{1/n}$. Elevando tutto alla potenza n-esima, si ha $\Big(\prod_{i=1}^{n} \lambda_i\Big) \geq (\sqrt{n})^n \det(\mathcal{L}(\mathbf{B}))$. Consideriamo ora i vettori ortogonalizzati $\mathbf{x}_i^*$ secondo la procedura di Gram-Schmidt e definiamo la trasformazione $T : \text{span}(\mathcal{L}(\mathbf{B})) \longrightarrow \text{span}(\mathcal{L}(\mathbf{B}))$

$$T\Big(\sum c_i \mathbf{x}_i^*\Big) = \sum \lambda_i c_i \mathbf{x}_i^*$$

che dilata ogni coordinata x_i^* di un fattore λ_i. Sia $S = \mathcal{B}(\mathbf{0}, 1) \cap \text{span}(\mathcal{L}(\mathbf{B}))$ la palla aperta n-dimensionale in $\text{span}(\mathcal{L}(\mathbf{B}))$. Se applichiamo T a S otteniamo un corpo convesso simmetrico $T(S)$ di volume

$$\text{vol}(T(S)) = \Big(\prod_i \lambda_i\Big)\text{vol}(S) \geq (\sqrt{n})^n \det(\mathcal{L}(\mathbf{B}))\text{vol}(S) = \text{vol}(\sqrt{n}S)\det(\mathcal{L}(\mathbf{B}))$$

dove $\sqrt{n}S$ è la palla di raggio $\sqrt{n}$. Il volume di $\sqrt{n}S$ è maggiore di 2^n perché $\sqrt{n}S$ contiene un ipercubo con spigolo di lunghezza 2. Quindi $\text{vol}(T(S)) > 2^n \det(\mathcal{L}(\mathbf{B}))$ e, per il teorema del corpo convesso di Minkowski, $T(S)$ contiene un punto del reticolo $\mathbf{y}$ diverso dall'origine. Dato che $\mathbf{y} \in T(S)$ dev'essere $\mathbf{y} = T(\mathbf{x})$ per qualche $\mathbf{x} \in S$. Dalla definizione di S otteniamo $||\mathbf{x}|| < 1$. Esprimiamo ora $\mathbf{x}$ e $\mathbf{y}$ nella base ortogonalizzata:

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{x}_i^*, \qquad \mathbf{y} = \sum_{i=1}^{n} \lambda_i c_i \mathbf{x}_i^*.$$

Dato che $\mathbf{y}$ è non nullo, qualche c_i non è zero. Sia k l'indice più grande tale che $c_i \neq 0$ e $k' \leq k$ l'indice più piccolo tale che $\lambda_{k'} = \lambda_k$. Notiamo che $\mathbf{y}$ è linearmente indipendente da $\mathbf{x}_1, \ldots, \mathbf{x}_{k'-1}$ perché $\mathbf{y} = \lambda_1 c_1 \mathbf{x}_1^* + \ldots + \lambda_k c_k \mathbf{x}_k^*$ con $\lambda_k c_k \neq 0$ e $\mathbf{x}_k^*$ è ortogonale allo span$\{\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}\}$. Mostriamo che $||\mathbf{y}|| < \lambda_k$.

$$\begin{aligned} ||\mathbf{y}||^2 &= \Big|\Big| \sum_{i \leq k} \lambda_i c_i \mathbf{x}_i^* \Big|\Big|^2 = \sum_{i \leq k} \lambda_i^2 c_i^2 ||\mathbf{x}_i^*||^2 \leq \sum_{i \leq k} \lambda_k^2 c_i^2 ||\mathbf{x}_i^*||^2 \\ &= \lambda_k^2 \Big|\Big| \sum_{i \leq k} c_i \mathbf{x}_i^* \Big|\Big|^2 = \lambda_k^2 ||\mathbf{x}||^2 < \lambda_k^2. \end{aligned}$$

Questo prova che $\mathbf{x}_1, ..., \mathbf{x}_{k'-1}, \mathbf{y}$ sono k' vettori linearmente indipendenti del reticolo di lunghezza strettamente minore di $\lambda_k = \lambda_{k'}$, contraddicendo la definizione di k-esimo minimo successivo λ_k. Pertanto $\left(\prod_{i=1}^n \lambda_i\right)^{1/n} < \sqrt{n} \det(\mathcal{L}(\mathbf{B}))^{1/n}$. □

2.4 Problemi computazionali

Il primo teorema di Minkowski fornisce un modo semplice di limitare la lunghezza λ_1 del vettore più corto non nullo nel reticolo $\mathcal{L}(\mathbf{B})$.
Tuttavia la dimostrazione del teorema di Minkowski non è costruttiva; ci dice che il vettore non nullo più corto esiste, ma non dà un metodo computazionale per trovarlo. Il problema di trovare un vettore del reticolo di lunghezza λ_1 è conosciuto come il problema del vettore più corto, o *Shortest Vector Problem* (SVP).

Definizione 2.16. (**Il problema del vettore più corto, SVP**) *Data una base* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$*, trovare un vettore non nullo del reticolo* $\mathbf{Bx}$ (*con* $\mathbf{x} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$) *tale che* $||\mathbf{Bx}|| \leq ||\mathbf{By}||$ *per ogni* $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$.

Un altro problema analogo per il quale non si conosce soluzione in un tempo polinomiale d'esecuzione è il problema del vettore più vicino, o *Closest Vector Problem*.

Definizione 2.17. (**Il problema del vettore più vicino, CVP**) *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$ *e fissato un vettore* $\mathbf{t} \in \mathbb{Z}^m$*, trovare il vettore del reticolo* $\mathbf{Bx}$ *più vicino a* $\mathbf{t}$*, in altre parole trovare un vettore* $\mathbf{x} \in \mathbb{Z}^n$ *tale che* $||\mathbf{Bx} - \mathbf{t}|| \leq ||\mathbf{By} - \mathbf{t}||$ *per ogni* $\mathbf{y} \in \mathbb{Z}^n$.

Entrambi i problemi, l'SVP e il CVP, possono essere considerati in diverse accezioni, di seguito elencate in ordine decrescente di difficoltà:

- il *Problema di Ricerca*: trovare un vettore non nullo del reticolo $\mathbf{x} \in \Lambda$ tale che $||\mathbf{x}||$ (rispettivamente $||\mathbf{x} - \mathbf{t}||$) è minimizzata;
- il *Problema di Ottimizzazione*: trovare il minimo di $||\mathbf{x}||$ (rispettivamente $||\mathbf{x} - \mathbf{t}||$) su $\mathbf{x} \in \Lambda \backslash \{\mathbf{0}\}$ (rispettivamente, $\mathbf{x} \in \Lambda$);
- il *Problema Decisionale*: dato un razionale $r > 0$, decidere se c'è un vettore non nullo del reticolo $\mathbf{x}$ tale che $||\mathbf{x}|| \leq r$ (rispettivamente, $||\mathbf{x} - \mathbf{t}|| \leq r$).

Dato che al momento non si conoscono algoritmi polinomiali per risolvere l'SVP e il CVP, vengono considerate versioni approssimate di questi problemi: gli algoritmi approssimati forniscono soluzioni che sono vicine a quella ottimale, a meno di un fattore specificato $\gamma > 1$. Tali versioni sono definite di seguito.

Definizione 2.18. (**Approximate SVP** o γ-SVP) *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$ *trovare un vettore non nullo del reticolo* $\mathbf{Bx}$ ($\mathbf{x} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$) *tale che* $||\mathbf{Bx}|| \leq \gamma \cdot ||\mathbf{By}||$ *per ogni* $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$.

Nelle applicazioni crittografiche il valore di γ sarà scelto in base alla dimensione del reticolo.

Definizione 2.19. (**Approximate CVP** o γ-CVP) *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$ *e fissato un vettore* $\mathbf{t} \in \mathbb{Z}^m$*, trovare un vettore* $\mathbf{x} \in \mathbb{Z}^n$ *tale che* $||\mathbf{Bx} - \mathbf{t}|| \leq \gamma ||\mathbf{By} - \mathbf{t}||$ *per ogni* $\mathbf{y} \in \mathbb{Z}^n$.

Un'ulteriore variante del SVP è il problema del vettore indipendente più corto, o *Shortest Independent Vector Problem* (SIVP), così definito:

Definizione 2.20. (γ**-SIVP**) *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$*, trovare* m *vettori linearmente indipendenti* $\mathbf{v}_1, ..., \mathbf{v}_m$ *non nulli del reticolo* $\mathcal{L}(\mathbf{B})$ *tali che* $||\mathbf{v}_i|| \leq \gamma \cdot \lambda_i$ *per ogni* $i \in \{1, ..., m\}$.

Anche il CVP può essere lievemente modificato nel problema di decodifica a distanza limitata, γ-*Bounded Distance Decoding Problem* (BDDP), o in una sua versione leggermente più debole, γ-BDDP', che possiamo trovare in [24].

Definizione 2.21. (γ**-BDDP**) *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$ *e fissato un vettore* $\mathbf{t} \in \mathbb{Z}^m$ *tale che* $\text{dist}(\mathbf{t}, \mathbf{B}) < \gamma \lambda_1(\mathcal{L}(\mathbf{B}))$*, trovare il vettore* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *più vicino a* $\mathbf{t}$.

Definizione 2.22. (γ-**BDDP**') *Data una base di vettori* $\mathbf{B} \in M_{m,n}(\mathbb{Z})$ *e fissato un vettore* $\mathbf{t} \in \mathbb{Z}^m$ *tale che* $\text{dist}(\mathbf{t}, \mathbf{B}) < \gamma\lambda_1(\mathcal{L}(\mathbf{B}))$*, trovare un vettore* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *tale che* $||\mathbf{v} - \mathbf{t}|| < \gamma\lambda_1(\mathcal{L}(\mathbf{B}))$.

In altre parole, il problema di decodifica a distanza limitata è il problema del vettore più vicino (γ-CVP) fissato un vettore che è stato già individuato vicino a un punto del reticolo, sperando così di ottenere proprio il vettore desiderato.
Eccetto il BDDP, i problemi suddetti sono considerati attualmente NP-hard per fattori di approssimazione molto piccoli, mentre per il BDDP diventa più difficile al crescere di γ e, per ora, risulta NP-hard per $\gamma > 1/\sqrt{2}$ [23]. Per tutti questi problemi i migliori algoritmi sono varianti dell'algoritmo di riduzione della base del reticolo conosciuto come LLL, dalle iniziali dei suoi creatori, Arjen Lenstra, Hendrik Lenstra e László Lovász (1982), si veda, ad esempio, nel libro di Cohen [7].
Per una trattazione più dettagliata riportiamo nell'Appendice B il contesto nel quale si collocano i problemi computazionali.

2.5 Reticolo-ideale

In termini generali se i reticoli hanno una struttura algebrica di gruppo, possiamo far corrispondere loro anche la nozione di ideale. Sia $f(x) \in \mathbb{Z}[x]$ un polinomio monico di grado n e consideriamo l'anello quoziente $\mathbb{Z}[x]/(f(x))$; utilizzando l'insieme usuale di rappresentanti $\{g(x) \bmod f(x) : g(x) \in \mathbb{Z}[x]\}$ e identificando i polinomi con i vettori (il vettore formato dai coefficienti del polinomio), l'anello quoziente $\mathbb{Z}[x]/(f(x))$ è isomorfo (come gruppo additivo) al reticolo $\mathbb{Z}^n$ e ogni ideale $I \subseteq \mathbb{Z}[x]/(f(x))$ individua un corrispondente sottoreticolo $\mathcal{L}(I) \subseteq \mathbb{Z}^n$. Diamo, allora, la seguente accezione di reticolo, che in inglese prende il nome di *ideal lattice*.

Definizione 2.23. *Un* ***reticolo-ideale*** $\mathcal{L}(B) \subseteq \mathbb{Z}^n$ *è un reticolo tale che* $\mathcal{L}(\mathbf{B}) = \{g(x) \bmod f(x) : g(x) \in I\}$ *per qualche polinomio monico* $f(x)$ *di grado* n *e per un ideale* $I \subseteq \mathbb{Z}[x]/(f(x))$.

Potendo interpretare $\mathbf{v} \in \mathbb{Z}[x]/(f(x))$ sia come elemento dell'anello dei polinomi sia come vettore dei coefficienti $\mathbf{v} \in \mathbb{Z}^n$, si ha allora che l'ideale $(\mathbf{v})$ generato da $\mathbf{v}$ corrisponde direttamente al reticolo generato dai vettori colonna $\{\mathbf{v}_i \leftarrow \mathbf{v} \times x^i \bmod f(x) : i \in [0, n-1]\}$, che chiamiamo *base di rotazione* (o *rotation basis*) del reticolo generato da $(\mathbf{v})$.

Definizione 2.24. *Una **base di rotazione** del reticolo generato da un ideale principale* $(\mathbf{v})$ *dell'anello dei polinomi* $\mathbb{Z}[x]/(f(x))$ *è data dai vettori colonna*

$$\{\mathbf{v}_i \leftarrow \mathbf{v} \times x^i \bmod \ f(x) : i \in [0, n-1]\}.$$

Esempio. Sia $f(x) = x^n + 1$ con n una potenza di 2, consideriamo

$$\mathbf{v} = \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix} = v_0 + v_1 x + \ldots + v_{n-1}x^{n-1} \in \mathbb{Z}[x]/(f(x)) \text{ allora}$$

$$\mathbf{v}_1 = \mathbf{v} \times x = \sum_{i=0}^{n-1} v_i x^i \times x = v_0 x + v_1 x^2 + \ldots - v_{n-1} \in \mathbb{Z}[x]/(f(x))$$

e analogamente per i successivi $\mathbf{v}_i$. La base di rotazione di $J = (\mathbf{v})$ può essere così scritta

$$\mathbf{B}_J = [\mathbf{v}, \mathbf{v}_1, \ldots, \mathbf{v}_{n-1}].$$

Osservazione 2.25. *L'ideale* $J = (\mathbf{v})$ *corrisponde naturalmente al reticolo* $\mathcal{L}(\mathbf{B}_J)$ *generato dalla base di rotazione* $\mathbf{B}_J$ *di* $(\mathbf{v})$.

Dimostrazione. Innanzitutto, per semplificare la notazione, poniamo $R = \mathbb{Z}[x]/(f(x))$. Ogni $\mathbf{w} \in (\mathbf{v})$ si può scrivere per definizione nella forma $\mathbf{w} = \mathbf{a} \times_R \mathbf{v}$ per qualche $\mathbf{a} \in R$ ed essendo $\times_R$ la moltiplicazione nell'anello R. Pertanto, associando gli elementi dell'anello a vettori formati dai coefficienti, possiamo scrivere che $\mathbf{w} = \sum_i a_i \mathbf{v}_i \in \mathcal{L}(\mathbf{B}_J)$. Viceversa se $\mathbf{w} \in \mathcal{L}(\mathbf{B}_J)$ allora è della forma $\mathbf{w} = \sum_i a_i \mathbf{v}_i$ per qualche intero a_i. Prendiamo $\mathbf{a} = \sum_i a_i x^i$ e abbiamo che $\mathbf{w} = \mathbf{a} \times_R \mathbf{v}$, che quindi appartiene all'ideale J. □

Parlando in termini più generali, un ideale $I \subset \mathbb{Z}[x]/(f(x))$ non è necessariamente principale. Supponiamo che l'ideale non sia principale, ma sia generato da $\mathbf{v}$ e $\mathbf{w}$. In questo caso l'ideale è rappresentato dal reticolo generato dalle colonne $\{\mathbf{v}_0, ..., \mathbf{v}_{n-1}, \mathbf{w}_0, ..., \mathbf{w}_{n-1}\}$, dove $\mathbf{z} \in \{\mathbf{v}_i, \mathbf{w}_i\}$ è il vettore associato a $\mathbf{z} \times x^i$, e i vettori nell'insieme sono linearmente indipendenti. Un algoritmo quale LLL potrà fornire una base per il reticolo associato a I che contenga solo vettori linearmente indipendenti.
Infine, al di là del nostro specifico contesto, consideriamo un generico dominio d'integrità R e il suo campo di frazioni K, possiamo così definire un **ideale frazionario** di R come un R-modulo $I \neq 0$, $I \subseteq K$, per cui esista un elemento non nullo $a \in R$ tale che $a \cdot I \subseteq R$. Da questa definizione possiamo introdurre la nozione di ideale inverso:

Definizione 2.26. *Un ideale frazionario I di R è detto* ***invertibile*** *se esiste un ideale frazionario J di R tale che $I \cdot J = R$, dove $I \cdot J = \left\{ \sum_{i=1}^{n} a_i b_i \ : \ a_i \in I, \ b_i \in J, \ n \in \mathbb{N} \right\}$. $J = I^{-1}$ è chiamato l'inverso di I.*

Nel nostro caso l'inverso di I, ideale frazionario di $\mathbb{Z}[x]/(f(x))$, sarà della forma $I^{-1} = \{\mathbf{x} \in \mathbb{Q}[x]/(f(x)) : \exists \mathbf{y} \in I, \mathbf{xy} \in \mathbb{Z}[x]/(f(x))\}$. Pertanto avremo bisogno anche dell'anello $\mathbb{Q}[x]/(f(x))$ e dall'ipotesi che $f(x)$ sia irriducibile cosicché tutti i termini non nulli abbiano l'inverso. In particolare un ideale principale $(\mathbf{v})$, con $\mathbf{v} \neq \mathbf{0}$, è invertibile e $(\mathbf{v})^{-1}$ è generato da $\mathbf{v}^{-1}$, l'inverso di $\mathbf{v}$ in $\mathbb{Q}[x]/(f(x))$. Per un ideale I generato da $\mathbf{v}_1, \ldots, \mathbf{v}_n$, l'inverso I^{-1} è l'intersezione di $(\mathbf{v}_1)^{-1}, \ldots, (\mathbf{v}_n)^{-1}$.
Nelle costruzioni che seguiranno (al fine di minimizzare la complessità dei circuiti di decodifica), utilizzeremo anelli di polinomi chiamati monogenici, o estensioni algebriche semplici, in quanto isomorfe a $\mathbb{Z}[\alpha]$, dove α è una radice di $f(x)$.
In termini più generali ricordiamo l'anello degli interi associato ad un campo di numeri: un campo di numeri K è un'estensione di grado finito del campo dei numeri razionali $\mathbb{Q}$ e l'anello degli interi di K è la chiusura integrale di $\mathbb{Z}$ in K:

$$\begin{aligned} \mathcal{O}_K &= K \cap \{\theta \in \mathbb{C} \ : \ p(\theta) = 0, \, p(x) \in \mathbb{Z}[x] \text{ monico}\} \\ &= \{\theta \in K \ : \ \theta^n + a_{n-1}\theta^{n-1} + \ldots + a_0 = 0, \, a_i \in \mathbb{Z}\}. \end{aligned}$$

L'anello degli interi ha una serie di proprietà che lo rendono interessante:

1. un ideale I non nullo di $\mathcal{O}_K$ è uno $\mathbb{Z}$-modulo libero di rango n;
2. il campo delle frazioni di $\mathcal{O}_K$ è K;
3. $\mathcal{O}_K/I$ è finito, dove I è un ideale non nullo di $\mathcal{O}_K$;
4. $\mathcal{O}_K$ è un dominio di Dedekind, pertanto
   - $\mathcal{O}_K$ è noetheriano,
   - ogni ideale primo non nullo di $\mathcal{O}_K$ è massimale,
   - $\mathcal{O}_K$ è integralmente chiuso nel suo campo delle frazioni,
   - ogni ideale non nullo di $\mathcal{O}_K$ può essere fattorizzato in modo unico come prodotto di ideali primi, a meno di permutare i fattori,
   - l'insieme degli ideali frazionari di $\mathcal{O}_K$ forma un gruppo rispetto alla moltiplicazione.

Capitolo 3

Fully Homomorphic Encryption Scheme using ideal lattices

Il nostro obiettivo ora è costruire un crittosistema pienamente omomorfo utilizzando anelli e reticoli descritti nel capitolo precedente; prima di tutto, però, inseriamo tale scoperta nel contesto della crittografia omomorfa e forniamo delle indicazioni su come svolgeremo il lavoro.

3.1 Preliminari

Questo capitolo illustra il primo approccio davvero innovativo e risolutivo alla ricerca di un crittosistema pienamente omomorfo; infatti, prima del lavoro di Gentry [12] del 2009, ci furono dei tentativi che andavano verso questa direzione, ma essi risultarono complessi e in grado di realizzare solo in parte l'omomorfia, ad esempio solo rispetto alla somma (Goldwasser-Micali del 1984, Naccache-Stern del 1998, Paillier del 1999) o solamente rispetto alla moltiplicazione (RSA del 1978, El Gamal del 1984). Un passo in avanti fu fatto da Boneh, Goh e Nissim nel 2005, il cui crittosistema (detto BGN) permette un numero illimitato di addizioni, ma solo una moltiplicazione. Il primo crittosistema ad essere stato pubblicato che risponde alla richiesta della piena omomorfia (sia rispetto all'addizione che alla moltiplicazione) è proprio quello che descriveremo in questo capitolo; anche il crittosistema basato sull'aritmetica modulare che abbiamo riportato nel Capitolo 1 è venuto alla luce solo in un secondo momento. La matematica soggiacente al metodo che presenteremo in questo capitolo è la Geometria dei Reticoli, che è stata scelta perché la complessità degli algoritmi di decodifica diventa molto bassa, specialmente se confrontata con crittosistemi quali RSA ed ElGamal.
Il metodo di Gentry può essere suddiviso in tre fasi:

1. si costruisce un crittosistema basato sui reticoli-ideali che sia parzialmente omomorfo, nel senso che sia in grado di valutare solo un numero limitato di operazioni sui dati codificati, corrispondenti a circuiti di profondità limitata (§3.2 - 3.4);

2. si comprime il circuito di decodifica del crittosistema ottenuto al punto 1, in modo che il crittosistema sia in grado di valutarne una sua versione leggermente aumentata e sia quindi bootstrappable (§3.6 - 3.7);

3. si fornisce una procedura di "refresh" per pulire i dati dagli errori accumulati dalle varie operazioni, che causerebbero una decodifica scorretta, e si ottiene un crittosistema pienamente omomorfo (§3.8).

Prima di procedere alla costruzione del crittosistema, spieghiamo per sommi capi l'idea sui cui esso si fonda. La codifica avviene tramite il concetto di vettore modulo una base, di cui ricordiamo la Definizione 2.10 data nel capitolo precedente: per un vettore $\mathbf{v} \in \mathbb{R}^m$, indichiamo con $\mathbf{v} \bmod \mathbf{B}$ l'unico vettore $\widetilde{\mathbf{v}} \in \mathcal{P}(\mathbf{B})$ tale che $\mathbf{v} - \widetilde{\mathbf{v}} \in \mathcal{L}(\mathbf{B})$.
Si tratta quindi di prendere un vettore, che sarà il messaggio da cifrare, aggiungergli un errore, dopodiché calcolarlo modulo una particolare base di un reticolo generato da un fissato ideale. Dati $\mathbf{v}$ e $\mathbf{B}$, $\mathbf{v} \bmod \mathbf{B}$ può essere calcolato come mostrato nell'Equazione 2.1:

$$\mathbf{v} \bmod \mathbf{B} = \mathbf{v} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{v} \rceil = \mathbf{B} \cdot \left(\mathbf{B}^{-1} \cdot \mathbf{v} - \lfloor \mathbf{B}^{-1} \cdot \mathbf{v} \rceil \right).$$

Per quanto riguarda la decodifica, essa consiste nel calcolare il vettore risultante dalla cifratura modulo un'altra base dello stesso reticolo-ideale usato per la codifica ed eliminare poi l'errore aggiunto inizialmente. Ciò è reso possibile dall'asimmetria tra la base pubblica e quella segreta, la prima viene detta "cattiva", mentre la seconda è una "buona" base. Con questi aggettivi intendiamo rispettivamente una base composta da vettori di lunghezza non minimale e lontani dall'essere ortogonali l'uno dall'altro (fatto che rende di difficile soluzione la ricerca del messaggio in chiaro sotteso), mentre con "buona" indichiamo una base composta da vettori corti e quasi ortogonali (fatto che permette di risolvere facilmente il problema). In particolare, una scelta possibile consiste nel porre la base pubblica uguale alla forma normale hermitiana (si vedano le Definizioni 2.6 e 2.7) della base segreta, e quest'ultima uguale ad una base di rotazione (Definizione 2.24) del reticolo-ideale iniziale.
Una misura che comunemente viene utilizzata per quantificare l'"ortogonalità" di una base di un reticolo $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n]$ è il *difetto di ortogonalità*, che confronta il prodotto della lunghez-

za dei vettori di base con il volume del parallelepipedo da essi definito: $\delta(\mathbf{B}) = \frac{\prod_{i=1}^{n} ||\mathbf{b}_i||}{\det \mathcal{L}(\mathbf{B})}$; se $\delta(\mathbf{B}) = 1$, allora i vettori della base $\mathbf{B}$ sono ortogonali.

3.2 Una costruzione iniziale

Nel nostro crittosistema iniziale $\mathcal{E}$ usiamo un anello fissato $R = \mathbb{Z}[x]/(f(x))$, dove $f(x)$ denota un polinomio, e consideriamo un suo ideale $I \subset R$. L'ideale I genera un reticolo di cui sia $\mathbf{B}_I$ una base. Utilizziamo l'algoritmo $\mathsf{IdealGen}(R, \mathbf{B}_I)$ per ottenere una base segreta $\mathbf{B}_J^{\mathsf{sk}}$ e una pubblica $\mathbf{B}_J^{\mathsf{pk}}$ dei reticoli generati da qualche ideale (variabile) J tale che $I + J = R$; I e J sono relativamente primi. Osserviamo che $\mathbf{B}_J^{\mathsf{sk}}$ può essere una base di un reticolo generato da un ideale frazionario che contiene J invece di essere generato esattamente da J.
Per la costruzione del crittosistema abbiamo bisogno di due nozioni: dati $\mathbf{t} \in R$ e $\mathbf{B}_M$ una base per un ideale $M \subset R$ osserviamo che

1. $\mathbf{t} \bmod \mathbf{B}_M$ è unico e può essere calcolato in modo efficiente, come abbiamo ricordato nel paragrafo precedente;

2. il laterale $\mathbf{t}+M$ ha un unico rappresentante "privilegiato" (*distinguished*, il rappresentante del laterale di lunghezza minima), ovvero calcolabile efficientemente rispetto alla base $\mathbf{B}_M$.

Indichiamo con $R \bmod \mathbf{B}_M$ l'insieme dei rappresentanti privilegiati di $\mathbf{r} + M$, $\mathbf{r} \in R$, rispetto alla particolare base $\mathbf{B}_M$ del reticolo generato da M. In questo schema lo spazio dei messaggi in chiaro $\mathcal{M}$ sarà un sottoinsieme di $R \bmod \mathbf{B}_I$.
Inoltre dobbiamo introdurre un altro algoritmo, $\mathsf{Samp}(\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J)$, che estragga dei campioni dal laterale $\mathbf{x}+I$. Lo applicheremo al laterale $\mathbf{m}+I$, dove $\mathbf{m}$ è il messaggio in chiaro; il risultato verrà poi ridotto modulo la base pubblica $\mathbf{B}_J^{\mathsf{pk}}$. Precisiamo, infine, che $\mathsf{Evaluate}$ prende come input un circuito C le cui porte svolgono le operazioni modulo $\mathbf{B}_I$; per esempio, una porta di addizione $\mathsf{Add}_{\mathbf{B}_I}$ prende due elementi di $R \bmod \mathbf{B}_I$ e fornisce un terzo termine in $R \bmod \mathbf{B}_I$ che è uguale alla somma dei primi due modulo I.

Avremo poi le seguenti funzioni:
$\mathsf{KeyGen}(R, \mathbf{B}_I)$. Prende come input un anello R e una base $\mathbf{B}_I$ del reticolo generato da un ideale I di R; tramite $\mathsf{IdealGen}(R, \mathbf{B}_I)$, fornisce $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}})$, basi di un reticolo generato da un ideale J di R: $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}}) \xleftarrow{\mathcal{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$.

La chiave pubblica pk include $R, \mathbf{B}_I, \mathbf{B}_J^{\mathsf{pk}}$ e Samp, mentre la chiave segreta sk contiene $\mathbf{B}_J^{\mathsf{sk}}$.

$\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m})$. Prende in input la chiave pubblica pk e un testo in chiaro $\mathbf{m} \in \mathcal{M}$, pone $\mathbf{c}' \leftarrow \mathsf{Samp}(\mathbf{m}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathsf{pk}})$, quindi $\mathbf{c}' = \mathbf{m} + \mathbf{i}$, con $\mathbf{i} \in I$. L'output risultante è $\mathbf{c} \leftarrow \mathbf{c}' \bmod \mathbf{B}_J^{\mathsf{pk}}$.

$\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$. A partire dalla chiave segreta sk e dal testo cifrato $\mathbf{c}$ dà come risultato $\mathbf{m} \leftarrow (\mathbf{c} \bmod \mathbf{B}_J^{\mathsf{sk}}) \bmod \mathbf{B}_I$.

$\mathsf{Evaluate}(\mathsf{pk}, C, \mathbf{c}_1, ..., \mathbf{c}_n)$. Prende in input la chiave pubblica pk, un circuito C in qualche insieme $\mathcal{C}_{\mathcal{E}}$ di circuiti permessi formati da porte $\mathsf{Add}_{\mathbf{B}_I}$ e $\mathsf{Mult}_{\mathbf{B}_I}$ e un insieme di testi cifrati $\mathbf{c}_1, ..., \mathbf{c}_n$. Applica il circuito ai testi cifrati e fornisce un ulteriore testo cifrato $\mathbf{c}$:
$\mathbf{c} \leftarrow g(C)(\mathbf{c}_1, ..., \mathbf{c}_n) \bmod \mathbf{B}_J^{\mathsf{pk}}$, dove $g(C)$ è il circuito generalizzato di C (si veda la Definizione 3.1).

$\mathsf{Add}_{\mathbf{B}_I}(\mathsf{pk}, \mathbf{c}_1, \mathbf{c}_2)$. Dati i testi cifrati $\mathbf{c}_1, \mathbf{c}_2$ dà come output $\mathbf{c}_1 + \mathbf{c}_2 \bmod \mathbf{B}_J^{\mathsf{pk}}$.

$\mathsf{Mult}_{\mathbf{B}_I}(\mathsf{pk}, \mathbf{c}_1, \mathbf{c}_2)$. Dati i testi cifrati $\mathbf{c}_1, \mathbf{c}_2$ dà come output $\mathbf{c}_1 \times \mathbf{c}_2 \bmod \mathbf{B}_J^{\mathsf{pk}}$.

Consideriamo ora la correttezza di tale crittosistema. La dimostrazione può essere suddivisa in due parti: la prima mostra che la decodifica è corretta per un testo cifrato fornito da $\mathsf{Encrypt}$, mentre la seconda mostra l'esattezza della decodifica di un testo risultante da $\mathsf{Evaluate}$.

La correttezza della decodifica di un testo fornito da $\mathsf{Encrypt}$

Volendo semplificare, la decodifica funziona grazie al fatto che la chiave segreta $\mathbf{B}_J^{\mathsf{sk}}$ genera un parallelepipedo $\mathcal{P}(\mathbf{B}_J^{\mathsf{sk}})$ che è grande abbastanza da permettere in tempo ragionevole di risolvere il problema di decodifica a distanza limitata (BDDP).
Un testo cifrato è della forma $\mathbf{c} = \mathbf{c}' + \mathbf{j}$ per qualche $\mathbf{j} \in J$. Dato che $\mathbf{B}_J^{\mathsf{sk}}$ è una base del reticolo generato dall'ideale J, $\mathbf{j} \in J$ può essere identificato con un elemento del reticolo e scritto come $\mathbf{j} = \mathbf{B}_J^{\mathsf{sk}} \cdot \mathbf{a}$, dove $\mathbf{a}$ è un vettore a coefficienti interi; pertanto

$$\mathbf{c} = \mathbf{c}' + \mathbf{B}_J^{\mathsf{sk}} \cdot \mathbf{a}.$$

La procedura di decodifica riduce $\mathbf{c}$ modulo $\mathbf{B}_J^{\mathsf{sk}}$:

$$\begin{aligned}\mathbf{c} \bmod \mathbf{B}_J^{\mathsf{sk}} &= \mathbf{c} - \mathbf{B}_J^{\mathsf{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c} \rceil = \mathbf{B}_J^{\mathsf{sk}} \cdot \big((\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c} - \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c} \rceil \big) \\ &= \mathbf{B}_J^{\mathsf{sk}} \cdot \big((\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot (\mathbf{c}' + \mathbf{B}_J^{\mathsf{sk}} \cdot \mathbf{a}) - \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot (\mathbf{c}' + \mathbf{B}_J^{\mathsf{sk}} \cdot \mathbf{a}) \rceil \big) \\ &= \mathbf{B}_J^{\mathsf{sk}} \cdot \big((\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' + \mathbf{a} - \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' + \mathbf{a} \rceil \big) \\ &= \mathbf{B}_J^{\mathsf{sk}} \cdot \big((\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' - \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' \rceil \big),\end{aligned}$$

dove l'ultima uguaglianza discende dal fatto che i coefficienti di $\mathbf{a}$ sono interi. Assumendo che $||\mathbf{c}'||$ sia minore di $1/(2||(\mathbf{B}_J^{\mathsf{sk}})^{-1}||)$, ogni coefficiente di $(\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}'$ risulta minore di $1/2$ in valore assoluto (come vedremo più dettagliatamente nel Lemma 3.14). Segue che $(\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' - \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' \rceil = (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}'$, da cui si ottiene

$$\mathbf{c} \bmod \mathbf{B}_J^{\mathsf{sk}} = \mathbf{B}_J^{\mathsf{sk}} \cdot \big((\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c}' \big) = \mathbf{c}'.$$

Da $\mathbf{c}'$ è facile estrarre il messaggio originale riducendolo modulo $\mathbf{B}_I$, dato che $\mathbf{c}' = \mathbf{m} + \mathbf{i}$ con $\mathbf{i} \in I$. Quindi l'algoritmo di decodifica è corretto se la chiave segreta è scelta adeguatamente.

La correttezza della decodifica di un testo fornito da Evaluate

Per mostrare la correttezza dell'algoritmo Evaluate abbiamo bisogno prima di dare qualche altra definizione.

Definizione 3.1. (**Circuito generalizzato**) *Sia C un circuito che opera* mod $\mathbf{B}_I$*. Definiamo il circuito generalizzato $g(C)$ di C come il circuito in cui le operazioni date dalle porte* $\mathsf{Add}_{\mathbf{B}_I}$ *e* $\mathsf{Mult}_{\mathbf{B}_I}$ *sono sostituite dall'addizione $'+'$ e la moltiplicazione $'\times'$ dell'anello R.*

Definizione 3.2. (X_{Enc}, X_{Dec}) *Sia X_{Enc} l'immagine di* Samp*. Possiamo osservare che tutti i testi cifrati forniti da* Encrypt *stanno in $X_{\mathsf{Enc}} + J$. Sia X_{Dec} uguale a $R \bmod \mathbf{B}_J^{\mathsf{sk}}$, ossia i rappresentanti privilegiati dei laterali di J rispetto alla base segreta $\mathbf{B}_J^{\mathsf{sk}}$.*

Definizione 3.3. (**Circuiti permessi**) *Sia*

$$\mathcal{C}_{\mathcal{E}}{}' = \{C : \forall (\mathbf{x}_1, ..., \mathbf{x}_t) \in X_{\mathsf{Enc}}^t, g(C)(\mathbf{x}_1, ..., \mathbf{x}_t) \in X_{\mathsf{Dec}}\}$$

l'insieme dei circuiti che operano mod $\mathbf{B}_I$ *che, quando generalizzati, danno un output sempre in X_{Dec} se gli input sono in X_{Enc}. Il valore t dipende naturalmente dal circuito C. Se $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}}{}'$ diciamo che $\mathcal{C}_{\mathcal{E}}$ è un insieme di circuiti permessi.*

Definizione 3.4. (**Testo cifrato valido**) *c è un testo cifrato valido rispetto alla chiave pubblica* pk *del crittosistema* $\mathcal{E}$ *e ai circuiti permessi* $\mathcal{C}_\mathcal{E}$ *se è ottenuto da* $\mathsf{Evaluate}(\mathsf{pk}, C, \mathbf{c}_1, ..., \mathbf{c}_n)$ *per qualche* $C \in \mathcal{C}_\mathcal{E}$*, dove ogni* $\mathbf{c}_i$ *appartiene all'immagine di* $\mathsf{Encrypt}$ *per* $i = 1, ..., n$*.*

Ora possiamo procedere a dimostrare il seguente teorema.

Teorema 3.5. *Assumiamo che* $\mathcal{C}_\mathcal{E}$ *sia un insieme di circuiti permessi contenente il circuito identità. Allora* $\mathcal{E}$ *è corretto per* $\mathcal{C}_\mathcal{E}$*, cioè* $\mathsf{Decrypt}$ *decifra correttamente i testi cifrati validi.*

Dimostrazione. Consideriamo l'insieme di testi cifrati $\mathbf{c}_1, ..., \mathbf{c}_t$, dove un testo $\mathbf{c}_k$ è per definizione uguale a $\mathbf{m}_k + \mathbf{i}_k + \mathbf{j}_k$ con $\mathbf{m}_k \in \mathcal{M}$, $\mathbf{i}_k \in I$, $\mathbf{j}_k \in J$ e $\mathbf{m}_k + \mathbf{i}_k \in X_{\mathsf{Enc}}$. Applicando un circuito C ai suddetti testi cifrati, abbiamo

$$\mathsf{Evaluate}(\mathsf{pk}, C, \mathbf{c}_1, ..., \mathbf{c}_n) = g(C)(\mathbf{c}_1, ..., \mathbf{c}_t) \bmod \mathbf{B}_J^{\mathsf{pk}} \in g(C)(\mathbf{m}_1 + \mathbf{i}_1, ..., \mathbf{m}_t + \mathbf{i}_t) + J.$$

Dato che $C \in \mathcal{C}_\mathcal{E}$ e $\mathbf{m}_k + \mathbf{i}_k \in X_{\mathsf{Enc}}$ abbiamo $g(C)(\mathbf{m}_1 + \mathbf{i}_1, ..., \mathbf{m}_t + \mathbf{i}_t) \in X_{\mathsf{Dec}}$ e quindi in $R \bmod \mathbf{B}_J^{\mathsf{sk}}$. Ora la decodifica dà

$$\begin{aligned} \mathsf{Decrypt}(\mathsf{sk}, \mathsf{Evaluate}(\mathsf{pk}, C, \mathbf{c}_1, ..., \mathbf{c}_t)) &= \left(g(C)(\mathbf{m}_1 + \mathbf{i}_1, ..., \mathbf{m}_t + \mathbf{i}_t) + J \bmod \mathbf{B}_J^{\mathsf{sk}}\right) \bmod \mathbf{B}_I \\ &= g(C)(\mathbf{m}_1 + \mathbf{i}_1, ..., \mathbf{m}_t + \mathbf{i}_t) \bmod \mathbf{B}_I \\ &= g(C)(\mathbf{m}_1, ..., \mathbf{m}_t) \bmod \mathbf{B}_I = C(\mathbf{m}_1 ...., \mathbf{m}_t). \end{aligned}$$

Abbiamo così provato che $\mathcal{E}$ è corretto per i circuiti permessi $\mathcal{C}_\mathcal{E}$. □

Adesso il nostro obiettivo consiste nel massimizzare quest'ultimo insieme $\mathcal{C}_\mathcal{E}$.

3.3 La sicurezza della costruzione astratta

Provvediamo a dare una dimostrazione della sicurezza semantica del crittosistema precedentemente descritto basata sul seguente problema astratto, l'*Ideal Coset Problem* o ICP.

Definizione 3.6. (**Problema del laterale di un ideale**, **ICP**) *Fissati* R*,* $\mathbf{B}_I$*, l'algoritmo* $\mathsf{IdealGen}$ *e un algoritmo* Samp_1 *che campiona* R*, si pone* $b \xleftarrow{\mathcal{R}} \{0, 1\}$ *e* $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}}) \xleftarrow{\mathcal{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$*.*
Se $b = 0$*, allora* $\mathbf{r} \xleftarrow{\mathcal{R}} \mathsf{Samp}_1(R)$ *e* $\mathbf{t} \leftarrow \mathbf{r} \bmod \mathbf{B}_J^{\mathsf{pk}}$*.*
Se $b = 1$*, allora* $\mathbf{t}$ *è scelto casualmente in modo uniforme da* $R \bmod \mathbf{B}_J^{\mathsf{pk}}$*.*
Il problema è trovare b *dato* $(\mathbf{t}, \mathbf{B}_J^{\mathsf{pk}})$*.*

Il problema in pratica chiede di decidere se $\mathbf{t}$ è uniforme modulo J o se è stato scelto secondo una distribuzione indotta da Samp_1. Per avere una dimostrazione basata sull'ICP definiamo prima di tutto l'algoritmo Samp.
Sia $\mathbf{s} \in I$ tale che l'ideale $(\mathbf{s})$ sia coprimo con J, (assumiamo per il momento che $\mathbf{s}$ possa essere efficientemente generato), allora $\mathsf{Samp}(\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathsf{pk}})$ è il seguente:

- Input: $\mathbf{x}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathsf{pk}}$,
- $\mathbf{r} \xleftarrow{\mathcal{R}} \mathsf{Samp}_1(R)$,dove Samp_1 campiona R, e calcola $\mathbf{x} + \mathbf{r} \times \mathbf{s}$, con $\mathbf{s} \in I$,
- Output: $\mathbf{x} + \mathbf{r} \times \mathbf{s} \in \mathbf{x} + I$.

Il valore $\mathbf{s}$ può essere un parametro fissato legato all'algoritmo Samp o generato dinamicamente da $\mathbf{B}_I$ e $\mathbf{B}_J^{\mathsf{pk}}$ in relazione a una distribuzione prescritta.

Teorema 3.7. *Supponiamo che esista un algoritmo $\mathcal{A}$ che violi la sicurezza semantica del crittosistema $\mathcal{E}$ con vantaggio ε quando usa* Samp*. Allora esiste un algoritmo $\mathcal{B}$ con lo stesso tempo di esecuzione di $\mathcal{A}$ che risolve l'ICP con vantaggio $\varepsilon/2$.*

Dimostrazione. L'avversario manda a $\mathcal{B}$ la coppia $(\mathbf{t}, \mathbf{B}_J^{\mathsf{pk}})$, $\mathcal{B}$ fissa $\mathbf{s}$ e gli altri componenti della chiave pubblica, sempre usando l'ICP. Quando $\mathcal{A}$ richiede la codifica di un messaggio tra $m_0, m_1 \in \mathcal{M}$, $\mathcal{B}$ fissa un bit $\beta \xleftarrow{\mathcal{R}} \{0,1\}$ e manda ad $\mathcal{A}$ la quantità $c \leftarrow m_\beta + \mathbf{t} \times \mathbf{s} \bmod \mathbf{B}_J^{\mathsf{pk}}$. Allora $\mathcal{A}$ ritorna un'ipotesi su β' e $\mathcal{B}$ congettura un $b' \leftarrow \beta \oplus \beta'$.
Se $b = 0$ affermiamo che la simulazione di $\mathcal{B}$ è perfetta; in particolare il testo cifrato ha la corretta distribuzione. Quando $b = 0$ abbiamo $\mathbf{t} = \mathbf{r} + j$, dove $\mathbf{r}$ è stato scelto in accordo con Samp_1 e $j \in J$; così $c \leftarrow m_\beta + \mathbf{t} \times \mathbf{s} = m_\beta + \mathbf{r} \times \mathbf{s} \bmod \mathbf{B}_J^{\mathsf{pk}}$ e il testo cifrato quindi è ben definito. In questo caso $\mathcal{A}$ ha vantaggio ε che si traduce in un vantaggio ε per $\mathcal{B}$.
Se $b = 1$ allora $\mathbf{t}$ è ottenuto in modo casuale uniforme modulo J: dato che l'ideale $(\mathbf{s})$ è coprimo con J, $\mathbf{t} \times \mathbf{s}$ è casuale modulo J e conseguentemente c, scelto casualmente, è indipendente da β e $c \in R \bmod \mathbf{B}_J^{\mathsf{pk}}$. In questo caso il vantaggio di $\mathcal{A}$ è 0 e il vantaggio di $\mathcal{B}$ è $\varepsilon/2$. □

3.4 La costruzione concreta

Quando implementiamo la costruzione astratta descritta sopra usando un anello di polinomi e i reticoli, gli insiemi X_{Enc} e X_{Dec} diventano sottoinsiemi di $\mathbb{Z}^n$. Caratterizziamo questi insiemi geometricamente.

Definizione 3.8. (r_{Enc} e r_{Dec}) *r_{Enc} è definito come il più piccolo valore tale che $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$, dove $\mathcal{B}(r)$ è la palla di raggio r, mentre r_{Dec} è il più grande valore tale che $X_{\mathsf{Dec}} \supseteq \mathcal{B}(r_{\mathsf{Dec}})$.*

Ora definiamo l'insieme dei circuiti permessi $\mathcal{C}_{\mathcal{E}}$ come segue:

$$\mathcal{C}_{\mathcal{E}} = \{C \ : \ \forall(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{B}(r_{\mathsf{Enc}})^t,\ g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{B}(r_{\mathsf{Dec}})\}.$$

$\mathcal{C}_{\mathcal{E}}$ è quindi definito come il massimo insieme $\mathcal{C}_{\mathcal{E}}{}'$ di circuiti permessi dati nella Definizione 3.3, dove però abbiamo sostituito X_{Enc} e X_{Dec} con $\mathcal{B}(r_{\mathsf{Enc}})$ e $\mathcal{B}(r_{\mathsf{Dec}})$, rispettivamente. Abbiamo $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}}{}'$. Caratterizziamo ora $\mathcal{C}_{\mathcal{E}}$, fissati r_{Enc} e r_{Dec}, cercando i limiti della lunghezza euclidea $||g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)||$. Questo si può realizzare limitando la lunghezza di $||\mathbf{u} + \mathbf{v}||$ e $||\mathbf{u} \times \mathbf{v}||$ in termini di $||\mathbf{u}||$ e $||\mathbf{v}||$ nel modo seguente: per l'addizione usiamo la disuguaglianza triangolare $||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}||$ per $\mathbf{u}, \mathbf{v} \in R$, mentre per la moltiplicazione possiamo provare che $||\mathbf{u} \times \mathbf{v}|| \leq \gamma_{\mathsf{Mult}}(R) \cdot ||\mathbf{u}|| \cdot ||\mathbf{v}||$, dove $\gamma_{\mathsf{Mult}}(R)$ è un fattore che dipende solo dall'anello R. Questo discende dal fatto che, moltiplicando i due polinomi $\mathbf{u}$ e $\mathbf{v}$ in R, si ottiene:

$$\mathbf{u} \times \mathbf{v} = \sum_{k=0}^{n} \sum_{j=0}^{k} u_j v_{k-j} x^{k+j},$$

dove $n = \deg(\mathbf{u}) + \deg(\mathbf{v})$, e possiamo definire γ come

$$\gamma(R) = \sup_{\mathbf{u},\mathbf{v} \neq \mathbf{0}} \frac{||\mathbf{u} \times \mathbf{v}||}{||\mathbf{u}|| \cdot ||\mathbf{v}||}.$$

Gentry in [12] usa queste limitazioni per dimostrare che il crittosistema parzialmente omomorfo $\mathcal{E}$ descritto in precedenza può valutare correttamente circuiti di profondità fino a $\log\log r_{\mathsf{Dec}} - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$.

Teorema 3.9. *Supponiamo che $r_{\mathsf{Enc}} \geq 1$ e che per il circuito C il numero di input accettabili (fan-in) dell'addizione sia $\gamma_{\mathsf{Mult}}(R)$ e della moltiplicazione sia 2. Inoltre la profondità di C sia al massimo*

$$\log\log r_{\mathsf{Dec}} - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}}).$$

Allora $C(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{B}(r_{\mathsf{Dec}})$ per ogni $\mathbf{x}_1, \ldots, \mathbf{x}_t \in \mathcal{B}(r_{\mathsf{Enc}})$.

Dimostrazione. Per un circuito di profondità d sia r_i un maggiorante della norma euclidea dei valori al livello i, con $i \in \{1, \ldots, d\}$, in altre parole i valori al livello i siano entro una palla di raggio r_i; si ha $r_d = r_{\mathsf{Enc}}$. Dalla disuguaglianza triangolare una porta di addizione (o sottrazione) al livello i dà come output qualche $\mathbf{v} \in R$ tale che $||\mathbf{v}|| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i$. Una porta

moltiplicativa al livello i fornisce come output qualche $\mathbf{v} \in R$ tale che $||\mathbf{v}|| \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$. In entrambi i casi $r_{i-1} \leq \gamma_{\mathsf{Mult}}(R) \cdot r_i^2$ e pertanto $r_0 \leq (\gamma_{\mathsf{Mult}}(R))^{2^d-1} \cdot (r_{\mathsf{Enc}})^{2^d} \leq (\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^{2^d}$. Allora deve risultare $(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^{2^d} \leq r_{\mathsf{Dec}}$ e la profondità d di C deve soddisfare questa disuguaglianza; la massima profondità si ha per $(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})^{2^d} = r_{\mathsf{Dec}}$, da cui $d = \log\log r_{\mathsf{Dec}} - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$. □

Semplificando ciò che viene enunciato dal teorema, possiamo affermare che, massimizzando la profondità dei circuiti che $\mathcal{E}$ può valutare correttamente, dobbiamo minimizzare $\gamma_{\mathsf{Mult}}(R)$ e r_{Enc} e massimizzare r_{Dec}.
Dal Teorema 3.9 appare perciò importante fissare $f(x)$ in modo tale che $R = \mathbb{Z}[x]/(f(x))$ abbia un basso valore di $\gamma_{\mathsf{Mult}}(R)$. I seguenti risultati mostrano che ci sono molti $f(x)$ per i quali il valore associato di $\gamma_{\mathsf{Mult}}(R)$ sia al più polinomiale in n.

Teorema 3.10. *Sia $f(x)$ un polinomio monico di grado n e sia $F(x) = x^n \cdot f(1/x)$ e $g(x) = F(x)^{-1} \bmod x^{n-1}$. Siano $u(x), v(x) \in \mathbb{Z}[x]/(f(x))$ e $\mathbf{u}, \mathbf{v}$ i vettori formati dai rispettivi coefficienti, allora $||\mathbf{u} \times \mathbf{v}|| \leq \gamma_{\mathsf{Mult}}(R) \cdot ||\mathbf{u}|| \cdot ||\mathbf{v}||$ per qualche*

$$\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot ||f|| \cdot ||g||).$$

Dimostrazione. Sia $t(x) \leftarrow u(x) \cdot v(x)$ il prodotto (non ridotto modulo $f(x)$) di $u(x)$ e $v(x)$ di grado $2n-2$. Possiamo dunque scrivere $t(x)$ come $t(x) = q(x)f(x) + s(x)$, dove $s(x) = t(x) \bmod f(x)$ è un polinomio di grado $n-1$ e $q(x)$ è un polinomio di grado $n-2$. Abbiamo $||\mathbf{u} \times \mathbf{v}|| = ||\mathbf{s}||$, in cui il secondo termine indica la norma euclidea del vettore formato dai coefficienti di $s(x)$. Notiamo che ogni coefficiente di $t(x)$, essendo il prodotto di qualche sottoinsieme di coefficienti di $\mathbf{u}$ e $\mathbf{v}$, deve avere norma minore di $||\mathbf{u}|| \cdot ||\mathbf{v}||$ (disuguaglianza di Cauchy-Schwarz). Allora

$$||\mathbf{t}||^2 = |t_0|^2 + \ldots + |t_{2n-2}|^2 \leq ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2 + \ldots + ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2 \leq 2n \cdot ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2,$$

da cui $||\mathbf{t}|| \leq \sqrt{2n} \cdot ||\mathbf{u}|| \cdot ||\mathbf{v}||$.
Sia $T(x) = x^{2n-2}t(1/x)$, $Q(x) = x^{n-2}q(1/x)$ e $S(x) = x^{2n-2}s(1/x)$. Allora $T(x) = Q(x)F(x) + S(x)$, dove T, Q, F sono tutti polinomi a coefficienti interi con lo stesso grado e norma di t, q, f rispettivamente. S, che ha la stessa norma di s, è divisibile per x^{n-1}, e ciò implica $Q(x) = T(x)g(x) \bmod x^{n-1}$; inoltre $Q(x)$ ha grado $n-2$, da cui $||Q|| \leq \sqrt{2n} \cdot ||T|| \cdot ||g||$.

Abbiamo pertanto che

$$\begin{aligned} ||\mathbf{u} \times \mathbf{v}|| = ||\mathbf{r}|| = ||S|| &\leq ||T|| + ||Q \cdot F|| \leq ||T|| + \sqrt{2n} \cdot ||Q|| \cdot ||F|| \\ &\leq ||T|| + 2n \cdot ||T|| \cdot ||g|| \cdot ||F|| = ||t|| \cdot (1 + 2n \cdot ||f|| \cdot ||g||) \\ &\leq ||\mathbf{u}|| \cdot ||\mathbf{v}|| \cdot \sqrt{2n} \cdot (1 + 2n \cdot ||f|| \cdot ||g||), \end{aligned}$$

e la tesi del teorema è dimostrata. □

Per trovare un anello $R = \mathbb{Z}[x]/(f(x))$ per cui $\gamma_{\mathsf{Mult}}(R)$ sia "piccolo", è sufficiente trovare un polinomio $f(x)$ per cui sia $F(x)$ che $F(x)^{-1} \bmod x^{n-1}$ abbiano norme "piccole". Questo ci dà libertà di scelta per $f(x)$; per esempio possiamo prendere $f(x)$ da un'ampia classe di polinomi tali che $f(x)$ abbia norma piccola e $f(x) = x^n - h(x)$ dove $h(x)$ è un polinomio di grado al massimo $(n+1)/2$. In questo caso, per $R = \mathbb{Z}[x]/(f(x))$, possiamo provare che $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot ||f||^2)$. Si può generalizzare quanto detto per $h(x)$ di grado al massimo $n - (n-1)/k$ per $k > 2$.

Teorema 3.11. *Sia $f(x) = x^n - h(x)$ dove $h(x)$ ha grado al massimo $n - (n-1)/k$ per $k \geq 2$. Allora, per $R = \mathbb{Z}[x]/(f(x))$, si ha*

$$\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot (\sqrt{(k-1)n}\, ||f||)^k).$$

Dimostrazione. Sia $F(x) = x^n \cdot f(1/x) = 1 - x^n \cdot h(1/x)$ e sia $H(x) = x^n \cdot h(1/x)$. Notiamo che $H(x)$ è divisibile per x^m per un intero $m \geq (n-1)/k$, dato che $h(x)$ ha grado al massimo $n - (n-1)/k$. Questo fatto implica che $1 - H(x)^k = 1 \bmod x^{k-1}$. Quindi

$$g(x) \leftarrow F(x)^{-1} = \frac{1}{1 - H(x)} = \frac{1 - (H(x))^k}{1 - H(x)} \bmod x^{n-1}$$

e abbiamo:

$$\begin{aligned} ||g(x)|| &\leq 1 + ||H|| + \ldots + ||H^{k-1}|| \leq 1 + ||H|| + \ldots + ((k-1)n)^{\frac{k-1}{2}} ||H||^{k-1} \\ &\leq 1 + ||f|| + \ldots + ((k-1)n)^{\frac{k-1}{2}} ||f||^{k-1} \leq \frac{(\sqrt{(k-1)n}\, ||f||)^k - 1}{(\sqrt{(k-1)n}\, ||f||) - 1}. \end{aligned}$$

Dato che $||f|| < (\sqrt{(k-1)n}\, ||f||) - 1$, abbiamo $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{2n} \cdot (1 + 2n \cdot (\sqrt{(k-1)n}\, ||f||)^k)$. □

Indubbiamente esistono polinomi $f(x)$ che non rientrano nella classe suddetta. Per esempio, siano $a_1, \ldots, a_k, b_1, \ldots, b_k$ polinomi tali che, per ogni i, $a_i = x^{r_i} - 1$ e $b_i = (1 - x^{r_i s_i})/(1 - x^{r_i})$ per qualche $\{r_i\}$,$\{s_i\}$ dove $r_i \cdot s_i \geq n-1$ e $r_i < n-1$. Allora, per ogni i, si ha $a_i b_i = 1 \bmod x^{n-1}$

e $||a_i||$ e $||b_i||$ sono entrambi piccoli. Potremmo fissare $F(x)$ e $g(x)$ prendendo un sottoinsieme $S \subseteq \{1, ..., k\}$ e ponendo $F(x) \leftarrow \prod_{i \in S} a_i \bmod x^{n-1}$ e $g(x) \leftarrow \prod_{i \in S} b_i \bmod x^{n-1}$. Le norme euclidee di F e g sarebbero piuttosto piccole, dato che le norme euclidee di a_i e b_i sono molto piccole.
Un caso semplice è porre $f(x) = x^n - 1$, per cui si ottiene l'anello $R = \mathbb{Z}[x]/(x^n - 1)$, $\gamma_{\mathsf{Mult}}(R) \leq \sqrt{n}$.

Lemma 3.12. *Siano $u(x), v(x) \in R = \mathbb{Z}[x]/(x^n - 1)$ e $\mathbf{u}$,$\mathbf{v}$ i corrispondenti vettori dei coefficienti, allora $||\mathbf{u} \times \mathbf{v}|| \leq \sqrt{n} \cdot ||\mathbf{u}|| \cdot ||\mathbf{v}||$.*

Dimostrazione. Sia $z(x)$ il polinomio risultante dalla moltiplicazione di $u(x)$ e $v(x)$, non ancora ridotto modulo $(x^n + 1)$, allora possiamo scrivere $z(x) = q(x)(x^n + 1) + r(x)$. Si ottiene che $||\mathbf{u} \times \mathbf{v}|| = ||\mathbf{r}||$, dove $||\mathbf{r}||$ denota la norma euclidea del vettore formato dai coefficienti di $r(x)$. Dato che l'i-esimo coefficiente r_i di $\mathbf{r}$ è il prodotto di qualche sottoinsieme di coefficienti di $\mathbf{u}$ e $\mathbf{v}$, risulta, per ogni $i = 0, \ldots, n-1$, $|r_i| \leq ||\mathbf{u}|| \cdot ||\mathbf{v}||$ (per la disuguaglianza di Cauchy-Schwarz) e si ha

$$\begin{aligned} ||\mathbf{u} \times \mathbf{v}||^2 = ||\mathbf{r}||^2 &= |r_0|^2 + \ldots + |r_{n-1}|^2 \\ &\leq \underbrace{||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2 + \ldots + ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2}_{n \text{ volte}} \\ &= n \cdot ||\mathbf{u}||^2 \cdot ||\mathbf{v}||^2. \end{aligned}$$

Possiamo quindi concludere che $||\mathbf{u} \times \mathbf{v}|| \leq \sqrt{n} \cdot ||\mathbf{u}|| \cdot ||\mathbf{v}||$. □

Preferiamo che $f(x)$ sia irriducibile, in modo tale che $K = \mathbb{Q}(x)/(f(x))$ risulti un campo e $\mathbb{Z}[x]/(f(x))$ erediti le proprietà dell'anello degli interi $\mathcal{O}_K$ corrispondente; inoltre, usando un polinomio irriducibile, si rende il sistema meno vulnerabile agli attacchi (per un'analisi della questione si veda [14]). Possiamo utilizzare $\mathbb{Z}[x]/(f(x))$ con $f(x) = (x^n - 1)/(x - 1)$, che è irriducibile quando n è primo, oppure con $f(x) = x^n + 1$, in cui n è una potenza di 2.
Dal Teorema 3.9 abbiamo come obiettivo di minimizzare r_{Enc}, rimanendo nei limiti della sicurezza. Ricordiamo che $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$ è l'immagine dell'algoritmo Samp usato in $\mathsf{Encrypt}$, dove la nostra dimostrazione sulla sicurezza vale quando $\mathsf{Samp}(\mathbf{B}_I, \mathbf{x})$ esegue $\mathbf{r} \leftarrow \mathsf{Samp}_1(R)$ e fornisce come output $\mathbf{x} + \mathbf{r} \times \mathbf{s}$ dove $\mathbf{s}$ è un generatore dell'ideale I. Sia l_{Samp_1} un maggiorante per la lunghezza di $\mathbf{r}$, abbiamo:

$$r_{\mathsf{Enc}} = \max\left\{||\mathbf{x} + \mathbf{r} \times \mathbf{s}||\right\} \leq n \cdot ||\mathbf{B}_I|| + \sqrt{n} \cdot l_{\mathsf{Samp}_1} \cdot ||\mathbf{B}_I||.$$

Per minimizzare r_{Enc}, possiamo scegliere $\mathbf{s}$ in modo che sia di norma piccola, ad esempio possiamo usare $\mathbf{s} = 2\mathbf{e}_1$, dove $\mathbf{e}_1$ è il vettore con il primo coefficiente pari a 1 e tutti gli altri nulli. Necessitiamo che l_{Samp_1} sia sufficientemente grande da aumentare la min-entropia di $\mathbf{t} \bmod \mathbf{B}_J^{\mathsf{pk}}$ nel problema ICP; per tale scopo basterà porre l_{Samp_1} polinomiale in n.

Osservazione 3.13. *In molti contesti legati alla sicurezza, tra cui il nostro, la capacità di ottenere una variabile casuale (in un singolo tentativo) è un'importante misura della qualità della variabile stessa. Questa capacità è racchiusa nella nozione di min-entropia:*
Una variabile aleatoria X ha ***min-entropia*** *pari a $k \in \mathbb{R}$, denotata con $H_\infty(X) = k$, se* $\max_x P[X = x] = 2^{-k}$.

Come esempio concreto possiamo porre $l_{\mathsf{Samp}_1} = n$: in tal caso avremmo un algoritmo Samp_1 che preleva vettori interi in $\mathcal{B}(l_{\mathsf{Samp}_1})$ casualmente in modo uniforme. Possiamo prendere anche r_{Enc} polinomiale in n. Notiamo che lo spazio dei messaggi in chiaro può essere delle dimensioni di $[R : I] = \det(I)$, che può essere esponenziale in n.
Per quanto riguarda r_{Dec}, osserviamo che esso dipende dalla base segreta dato che r_{Dec} è il raggio della sfera più grande centrata in $\mathbf{0}$ circoscritta da $\mathbf{B}_J^{\mathsf{sk}}$. Sempre per il Teorema 3.9, vogliamo massimizzare r_{Dec}, facendo in modo che il parallelepipedo $\mathcal{P}(\mathbf{B}_J^{\mathsf{sk}})$ sia grande abbastanza da contenere una grande sfera. Questa proprietà è formalizzabile nei termini della matrice inversa $(\mathbf{B}_J^{\mathsf{sk}})^{-1}$, la cui trasposta è una base (o un insieme di vettori linearmente indipendenti) del reticolo duale $\mathcal{L}(\mathbf{B}_J^{\mathsf{sk}})$.

Lemma 3.14. *Sia $\mathbf{B}$ la base di un reticolo e $\mathbf{B}^* = (\mathbf{B}^{-1})^T$ e sia r il raggio della sfera più grande, centrata in $\mathbf{0}$, circoscritta da $\mathcal{P}(\mathbf{B})$ (dove è possibile la tangenza). Allora $r = 1/(2 \cdot ||\mathbf{B}^*||)$.*
In particolare,

$$r_{\mathsf{Dec}} = \frac{1}{2 \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1})^T||}.$$

Supponiamo che $||\mathbf{t}|| < r_{\mathsf{Dec}}$; allora il modulo di ogni coefficiente di $\mathbf{B}^{-1} \cdot \mathbf{t}$ è, al massimo, pari a $1/2$.

Dimostrazione. Supponiamo che $||\mathbf{x}|| < 1/(2 \cdot ||\mathbf{B}^*||)$. Ogni coefficiente di $\mathbf{B}^{-1} \cdot \mathbf{x}$ è il prodotto interno di $\mathbf{x}$ con un vettore colonna di $\mathbf{B}^*$, e quindi ha norma al massimo $||\mathbf{x}|| \cdot ||\mathbf{B}^*|| < 1/2$. Questo implica che $\lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil = 0$, $\mathbf{x} = (\mathbf{x} \bmod \mathbf{B})$ e $\mathbf{x} \in \mathcal{P}(\mathbf{B})$.
Ora supponiamo che $||\mathbf{x}|| > 1/(2 \cdot ||\mathbf{B}^*||)$ e che $\mathbf{x}$ sia parallelo al vettore più lungo $\mathbf{b}_i$ in $\mathbf{B}^*$. Allora $|\langle \mathbf{b}_i, \mathbf{x} \rangle| > 1/2$, che implica $\lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil \neq 0$, $\mathbf{x} \neq (\mathbf{x} \bmod \mathbf{B})$ e $\mathbf{x} \notin \mathcal{P}(\mathbf{B})$. □

La rilevanza del Lemma 3.14 sta nel fatto che l'equazione di decodifica, $\mathbf{m} = \mathbf{c} - \mathbf{B}_J^{\mathsf{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c} \rceil \bmod \mathbf{B}_I$, è corretta quando c dista al massimo $r_{\mathsf{Dec}} = 1/(2 \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1})^T||)$ da un punto del reticolo in J. Ad esempio, possiamo generare in modo casuale un vettore $\mathbf{v}$, porre $\mathbf{B}_J^{\mathsf{sk}}$ uguale alla base di rotazione di $\mathbf{v}$ e $\mathbf{B}_J^{\mathsf{pk}}$ la forma normale hermitiana di $(\mathbf{v})$. Parlando in termini approssimativi, se $\mathbf{v}$ è generato come vettore prossimo ad essere parallelo a $\mathbf{e}_1 = (1, 0, \ldots, 0)$, allora la sfera più grande circoscritta da $\mathcal{P}(\mathbf{B}_\mathbf{v})$, dove $\mathbf{B}_\mathbf{v}$ è la base di rotazione di $\mathbf{v}$, avrà raggio r_{Dec} entro un fattore polinomiale di $\lambda_1(J)$.

Lemma 3.15. *Sia $s > 0$ e $t \geq 4n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s$, supponiamo che $\mathbf{v} \in t \cdot \mathbf{e}_1 + \mathcal{B}(s)$, ovvero $\mathbf{v}$ è nella palla di raggio s centrata in $t \cdot \mathbf{e}_1$, e che $\mathbf{B}$ sia la base di rotazione di $\mathbf{v}$. Allora $\mathcal{P}(\mathbf{B})$ circoscrive una palla di raggio almeno $t/4$.*

Dimostrazione. Per $i \in [0, n-1]$, siano $\mathbf{v}_i = \mathbf{v} \times x^i$ e $\mathbf{z}_i = \mathbf{v}_i - t \cdot \mathbf{e}_i$. Abbiamo

$$||\mathbf{z}_i|| = ||\mathbf{z}_0 \times x^i|| \leq \gamma_{\mathsf{Mult}}(R) \cdot ||\mathbf{z}_0|| \leq \gamma_{\mathsf{Mult}}(R) \cdot s;$$

in altre parole, $\mathbf{v}_i = t \cdot \mathbf{e}_1 + \mathbf{z}_i$ risulta prossimo ad essere parallelo ad $\mathbf{e}_i$ quando $\gamma_{\mathsf{Mult}}(R) \cdot s$ è molto più piccolo di t. Inoltre per ogni punto $\mathbf{a}$ sulla superficie di $\mathcal{P}(\mathbf{B})$ c'è un indice i tale che

$$\mathbf{a} = \pm\frac{1}{2} \cdot \mathbf{v}_i + \sum_{j \neq i} x_j \cdot \mathbf{v}_j$$

per $x_j \in [-1/2, 1/2]$. Quindi

$$|\langle \mathbf{a}, \mathbf{e}_i \rangle| \geq t/2 - n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s.$$

In particolare, $||\mathbf{a}|| \geq t/2 - n \cdot \gamma_{\mathsf{Mult}}(R) \cdot s \geq t/2 - t/4 \geq t/4$, da cui segue il lemma. □

3.5 La sicurezza del sistema concreto

Quando consideriamo la costruzione concreta, il problema del laterale di un ideale (ICP) diventa il (Decision) Bounded Distance Decoding Problem per i reticoli, o Decision BDDP.

Definizione 3.16. (**Decision BDDP per i reticoli**) *Fissati un anello di polinomi $R = \mathbb{Z}[x]/(f(x))$, l'algoritmo* $\mathsf{IdealGen}$ *che fornisce basi di un ideale in R e l'algoritmo* Samp_1 *che estrae campioni da R, si pone $b \xleftarrow{\mathcal{R}} \{0,1\}$ e $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}}) \xleftarrow{\mathcal{R}} \mathsf{IdealGen}(R, \mathbf{B}_I)$.*
Se $b = 0$ allora $\mathbf{r} \xleftarrow{\mathcal{R}} \mathsf{Samp}_1(R)$ e $\mathbf{t} \leftarrow \mathbf{r} \bmod \mathbf{B}_J^{\mathsf{pk}}$.
Se $b = 1$ allora $\mathbf{t}$ è scelto casualmente in modo uniforme da $R \bmod \mathbf{B}_J^{\mathsf{pk}}$.
Il problema è trovare b dato $(\mathbf{t}, \mathbf{B}_J^{\mathsf{pk}})$.

La difficoltà del Decision BDDP dipende in modo cruciale dall'algoritmo Samp_1 e da come è generato J, in particolare dal valore di $\lambda_1(J)$, che ricordiamo essere il vettore più corto del reticolo associato a J e di cui abbiamo fornito una limitazione tramite il Teorema del Corpo Convesso 2.14.
Per quanto riguarda Samp_1, osserviamo che se l'algoritmo fornisse sempre il vettore $\mathbf{0}$ o campionasse in accordo a qualche distribuzione con min-entropia molto bassa, allora il problema BDDP diventerebbe facile; mentre se la distribuzione avesse min-entropia alta (ad esempio se il vettore fornito da Samp_1 fosse campionato da una sfera di raggio n), il problema BDDP diventerebbe di difficile risoluzione (per la definizione di min-entropia si veda l'Osservazione 3.13).
In merito a come è generato J e al valore di $\lambda_1(J)$, possiamo notare che se $\lambda_1(J)/l_{\mathsf{Samp}_1} \geq 2^n$, ovvero se $\lambda_1(J)$ è molto più grande di l_{Samp_1}, l'algoritmo del piano più vicino di Babai può essere usato per trovare il vettore di J più vicino a $\mathbf{t}$ in un tempo polinomiale.

Babai ha introdotto due metodi per trovare il vettore più vicino a $\mathbf{t}$: una tecnica di arrotondamento (*rounding technique*) e l'algoritmo del piano più vicino (*nearest plane algorithm*).

Tecnica di arrotondamento. Data una base $\mathbf{B}$ di un reticolo, si calcola $\mathbf{u} = \mathbf{B}\lfloor\mathbf{B}^{-1}\mathbf{t}\rceil$, che risulta una possibile approssimazione del vettore più vicino a $\mathbf{t}$. Se la base è ridotta (non lontana dall'essere ortogonale, ad esempio tramite il processo di ortogonalizzazione di Gram-Schmidt o, meglio ancora, tramite l'algoritmo *lattice basis reduction*, LLL), Babai ha dimostrato che la soluzione fornita dista dalla risposta corretta al più una costante esponenziale della distanza minima del reticolo: $||\mathbf{u} - \mathbf{t}|| \leq (1 + 2n(9/2)^{n/2}) \min\{||\mathbf{z} - \mathbf{t}|| \, : \, \mathbf{z} \in \mathcal{L}(\mathbf{B})\}$, dove n è il rango di $\mathcal{L}(\mathbf{B})$ [11]. Tramite l'esempio mostrato in Figura 3.1 risulta più chiara la scelta di dare l'appellativo di "buona" ad una base con vettori corti e quasi ortogonali e di "cattiva" altrimenti.

Algoritmo del piano più vicino. Se la base ridotta del reticolo è $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, allora si trova $\lambda_n \in \mathbb{Z}$ tale che l'iperpiano $\lambda_n\mathbf{b}_n + \mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ sia il più vicino a $\mathbf{t}$; successivamente si sostituisce $\mathbf{t}$ con $\mathbf{t} - \lambda_n\mathbf{b}_n$ e si ripete ricorsivamente lo stesso metodo alla base $\{\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}\}$. Il vettore del reticolo risultante è $\mathbf{u} = \sum_i \lambda_i\mathbf{b}_i$ e Babai ha mostrato che $||\mathbf{u} - \mathbf{t}|| \leq 2(4/3)^{n/2} \min\{||\mathbf{z} - \mathbf{t}|| \, : \, \mathbf{z} \in \mathcal{L}(\mathbf{B})\}$.

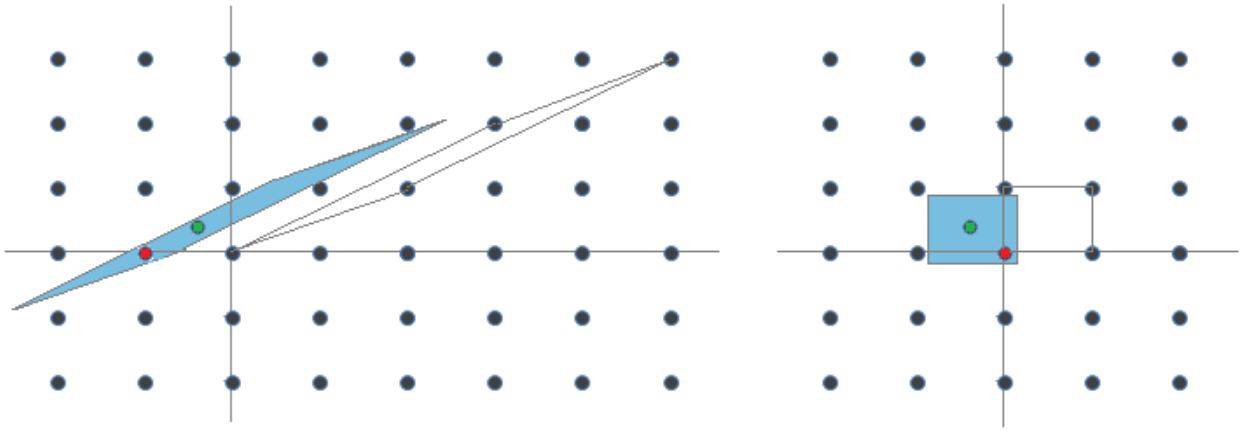

Figura 3.1: Sinistra: parallelepipedo centrato in $(-0.4, 0.4)$ corrispondente alla base $\{(3,2);(2,1)\}$. Destra: parallelepipedo centrato in $(-0.4, 0.4)$ corrispondente alla base $\{(1,0);(0,1)\}$.

Abbiamo osservato che r_{Enc} è al massimo $n \cdot ||\mathbf{B}_I|| + \sqrt{n} \cdot l_{\mathsf{Samp}_1} \cdot ||\mathbf{B}_I||$, dove $\mathbf{B}_I$ è scelto in modo che $||\mathbf{B}_I||$ sia polinomiale in n. In breve, r_{Enc} può essere più grande di l_{Samp_1} solo di un fattore polinomiale; mentre per quanto riguarda r_{Dec}, esso è minore di $\lambda_1(J)$ sempre solo di un fattore polinomiale, per una adeguata base segreta $\mathbf{B}_J^{\mathsf{sk}}$. Quindi possiamo fare in modo che $r_{\mathsf{Dec}}/r_{\mathsf{Enc}}$ sia entro un fattore polinomiale di $\lambda_1(J)/l_{\mathsf{Samp}_1}$, dove quest'ultimo è essenzialmente un fattore approssimativo del Decision BDDP. Gentry in [12] ha suggerito di porre $r_{\mathsf{Dec}} = 2^{O(\sqrt{n})}$ e $r_{\mathsf{Enc}} = \text{poly}(n)$, che soddisfano le osservazioni sulla sicurezza descritte sopra. In particolare, quando $r_{\mathsf{Dec}} = 2^{n^{c_1}}$ e $\gamma(R) \cdot r_{\mathsf{Enc}} = 2^{n^{c_2}}$, grazie ai Teoremi 3.5 e 3.9, il crittosistema potrà valutare correttamente circuiti di profondità $(c_1 - c_2) \log n$.

3.6 Modifiche al sistema parzialmente omomorfo

A questo punto, avendo delineato il nostro sistema parzialmente omomorfo $\mathcal{E}$, possiamo chiederci se sia bootstrappable. Prima di tutto, però, descriviamo due modifiche da apportare al crittosistema per abbassare la complessità di decodifica senza ridurre sostanzialmente la profondità che il sistema può valutare.
La prima modifica riguarda la chiave segreta così che l'equazione della decodifica si semplifichi da

$$m = c - \mathbf{B}_J^{\mathsf{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot c \rceil \bmod \mathbf{B}_I$$

a

$$m = c - \lfloor \mathbf{v}_J^{\mathsf{sk}} \times c \rceil \bmod \mathbf{B}_I$$

dove $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$.

Prima di descrivere la modifica, è utile capire la relazione tra il duale di un reticolo (una buona base usata come chiave di decodifica) e l'inverso di un reticolo (un vettore usato come chiave di decodifica nell'equazione modificata).

3.6.1 La relazione tra il duale e l'inverso di un reticolo

Ricordiamo la Definizione 2.8 del duale J^* di un reticolo-ideale J,

$$J^* = \{\mathbf{x} \in \mathbb{R}^n \,:\, \forall \mathbf{v} \in J,\, \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\},$$

e la Definizione 2.26 dell'inverso J^{-1} in $R = \mathbb{Z}[x]/(f(x))$, interpretando J come ideale di R,

$$J^{-1} = \{\mathbf{x} \in \mathbb{Q}[x]/(f(x)) \,:\, \forall \mathbf{v} \in J,\, \mathbf{x} \times \mathbf{v} \in R\}.$$

Se $\mathbf{B}_J$ è la base di rotazione di $J = (\mathbf{v})$, allora l'inverso $J^{-1} = (1/\mathbf{v})$ è generato dalla base di rotazione di $1/\mathbf{v}$, le colonne di $\mathbf{B}_J^{-1}$, e quindi $J^{-1} = \mathcal{L}(\mathbf{B}_J^{-1})$. Per quanto riguarda il duale di J, esso è generato dall'inversa della trasposta di $\mathbf{B}_J$ (si veda il Teorema 2.9).

Lemma 3.17. *Siano* $\mathbf{B}$ *la base di rotazione di* J *e* $\mathbf{B}^*$ *la sua trasposta inversa. Allora* $||\mathbf{B}^*|| \cdot \sqrt{n} \geq ||\mathbf{B}^{-1}|| \geq ||\mathbf{B}^*||/\sqrt{n}$. *In particolare, se* $\mathbf{B}_J^{\mathsf{sk}}$ *è una base di rotazione, abbiamo*

$$\frac{1}{2\sqrt{n} \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1}||} \leq r_{\mathsf{Dec}} \leq \frac{\sqrt{n}}{2 \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1}||}.$$

Dimostrazione. Sia b_{ij} il coefficiente di valore assoluto maggiore nella matrice $\mathbf{B}^{-1}$, allora

$$||\mathbf{B}^{-1}|| \geq b_{ij} \geq ||\mathbf{B}^*||/\sqrt{n} \quad \text{e} \quad ||\mathbf{B}^*|| \geq b_{ij} \geq ||\mathbf{B}^{-1}||/\sqrt{n}.$$

Usando il Lemma 3.14 ne consegue

$$\frac{1}{2\sqrt{n} \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1}||} \leq r_{\mathsf{Dec}} \leq \frac{\sqrt{n}}{2 \cdot ||(\mathbf{B}_J^{\mathsf{sk}})^{-1}||}. \qquad \square$$

Ci chiediamo ora se è possibile caratterizzare questa relazione tra il duale e l'inverso di un reticolo qualsiasi: per esempio, dato un vettore di piccola lunghezza in J^{-1} possiamo trovare una base di norma piccola di J^*? O dato un vettore corto in J^* possiamo fornire come output una base di norma piccola per J^{-1}? La risposta è affermativa; in particolare la motivazione alla

prima domanda risiede nel Lemma 3.17. Sia $\mathbf{B}_J$ una base di J con vettori colonna $\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}$; se $\mathbf{v}$ è un vettore corto in J^{-1} e $\mathbf{B}_\mathbf{v}$ è una base di rotazione, allora $\mathbf{v} \times \mathbf{u}_i \in R$ per ogni i e $\mathbf{B}_\mathbf{v} \cdot \mathbf{B}_J$ risulta una matrice intera. Ciò implica che le righe di $\mathbf{B}_\mathbf{v}$ formano un insieme indipendente in J^*, ma il vettore riga di lunghezza maggiore non può superare la lunghezza del vettore colonna più lungo come nel Lemma 3.14. La seconda questione, ovvero se possiamo generare una base corta di J^{-1} da un vettore corto in J^*, è risolvibile mediante il seguente lemma.

Lemma 3.18. *Sia $\mathbf{w} \in J^*$, dove J^* è il duale di un reticolo J, e sia*

$$\mathbf{v} = \sum_{i=0}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1},$$

allora $\mathbf{v} \in J^{-1}$. Se $\mathbf{B}_\mathbf{v}$ è la base di rotazione di $\mathbf{v}$ allora $||\mathbf{B}_\mathbf{v}|| \leq \sqrt{n} \cdot ||f|| \cdot ||\mathbf{w}||$. Questo si applica anche quando J è un ideale frazionario.

La dimostrazione si basa sull'idea di prendere $\mathbf{w} \in J^*$, porlo nell'ultima riga di una matrice $n \times n$ per poi completarla per avere la base di rotazione di un vettore in J^{-1}. Insieme il vettore $\mathbf{w}$ e il polinomio $f(x)$ dettano come deve essere il resto della matrice.

Dimostrazione. Affermiamo che l'ultima riga di $\mathbf{B}_\mathbf{v}$ sia $(w_0, w_1, \ldots, w_{n-1})$. Denotiamo le colonne di $\mathbf{B}_\mathbf{v}$ con $\mathbf{v}^{(k)} = \mathbf{v} \cdot x^k \bmod f(x)$. Vogliamo mostrare che

$$\mathbf{v}^{(k)} = \sum_{i=k}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=0}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k},$$

da cui il coefficiente di x^{n-1} in $\mathbf{v}^{(k)}$ risulta w_k. Procediamo per induzione su k: se $k = 0$

$$\mathbf{v}^{(0)} = \sum_{i=0}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1} = \mathbf{v} \cdot x^0 \bmod f(x),$$

che è verificato dalle ipotesi. Assumiamo ora che la tesi sia vera per $k-1$; abbiamo

$$\begin{aligned}
\mathbf{v}^{(k)} &= x\left(\sum_{i=k-1}^{n-1} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k-1} - \sum_{i=0}^{k-2} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k-1} \right) \bmod f(x) \\
&= \left(\sum_{i=k}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=1}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k} \right) \bmod f(x) \\
&= \sum_{i=k}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=1}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k} - (f_n \cdot w_{k-1}) \cdot f(x) \\
&= \sum_{i=k}^{n} x^i \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=1}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k} - \sum_{i=0}^{n} x^i \cdot w_{k-1} \cdot f_i
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=k}^{n} x^i \left(- w_{k-1} \cdot f_i + \sum_{j=i}^{n} f_j \cdot w_{j-i-1+k} \right) \\
&\qquad - \sum_{i=1}^{k-1} x^i \left(w_{k-1} \cdot f_i + \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k} \right) - w_{k-1} \cdot f_0 \\
&= \sum_{i=k}^{n} x^i \sum_{j=i+1}^{n} f_j \cdot w_{j-i-1+k} - \sum_{i=0}^{k-1} x^i \sum_{j=0}^{i} f_j \cdot w_{j-i-1+k},
\end{aligned}
$$

come richiesto.

Per mostrare che $\mathbf{v} \in J^{-1}$, basta provare che $\mathbf{z} \leftarrow \mathbf{v} \times \mathbf{x} \in R$ per ogni $\mathbf{x} \in J$. Siano $\mathbf{B_x}$ e $\mathbf{B_z}$ le rispettive basi di rotazione di $\mathbf{x}$ e $\mathbf{z}$, sappiamo che $\mathbf{B_z} = \mathbf{B_v} \cdot \mathbf{B_x}$ e che l'ultima riga di $\mathbf{B_z}$ ha valori interi, dato che questa riga è $\mathbf{w} \cdot \mathbf{B_x}$ e $\mathbf{w} \in J^*$ genera un prodotto intero con tutti i vettori in J (anche le colonne di $\mathbf{B_x}$).

Supponiamo per assurdo che $\mathbf{z}$ non sia un vettore intero, in particolare che i^* sia l'indice maggiore per cui z_{i^*}, il coefficiente corrispondente di $\mathbf{z}$, non sia intero. Consideriamo $\mathbf{z}^{(n-i^*-1)} \leftarrow x^{n-i^*-1} \cdot \mathbf{z} \bmod f(x)$, che è un vettore colonna in $\mathbf{B_z}$. Possiamo scrivere $\mathbf{z}^{(n-i^*-1)} = x^{n-i^*-1} \cdot \mathbf{z} - a(x) \cdot f(x)$, dove $a(x)$ è un polinomio a coefficienti interi. Abbiamo che in $x^{n-i^*-1} \cdot \mathbf{z}$ i coefficienti dei termini da x^n a x^{2n-i^*-2} sono interi, mentre il coefficiente di x^{n-1} in $x^{n-i^*-1} \cdot \mathbf{z}$ non è intero, dato che z_{i^*} non lo è. Noi sappiamo che $\mathbf{z}^{(n-i^*-1)}$ differisce da $x^{n-i^*-1} \cdot \mathbf{z}$ di un polinomio intero, quindi anche il coefficiente di x^{n-1} in $\mathbf{z}^{(n-i^*-1)}$ non può essere intero, ma avevamo stabilito che l'ultima riga di $\mathbf{B_z}$ era intera, generando così una contraddizione. Pertanto $\mathbf{z} \in R$ e $\mathbf{v} \in J^{-1}$.

Infine, per quanto riguarda $||\mathbf{B_v}||$, sappiamo che ogni elemento della matrice è il prodotto interno di due vettori, uno a coefficienti in $\{f_0, \ldots, f_n\}$ e l'altro a coefficienti in $\{w_0, \ldots, w_{n-1}\}$ a meno del segno. Allora il modulo di ogni elemento di $\mathbf{B_v}$ è al massimo $||f|| \cdot ||\mathbf{w}||$, che implica $||\mathbf{B_v}|| \leq \sqrt{n} \cdot ||f|| \cdot ||\mathbf{w}||$. □

Come immediato corollario, otteniamo una limitazione del determinante del reticolo $\mathcal{L}(\mathbf{B_v})$, generato da J^{-1} con base $\mathbf{B_v}$, in termini del determinante del reticolo $\mathcal{L}(\mathbf{B})$, generato da J con base $\mathbf{B}$, e un estremante per $\lambda_n(\mathcal{L}(\mathbf{B_v}))$ in relazione a $\lambda_n(\mathcal{L}(\mathbf{B}))$.

Lemma 3.19. *Sia J un ideale (anche frazionario) di $R = \mathbb{Z}[x]/(f(x))$ allora $\lambda_n(\mathcal{L}(\mathbf{B_v})) \leq \sqrt{n} \cdot ||f|| \cdot \lambda_1(\mathcal{L}(\mathbf{B}^*)) \leq n \cdot ||f||/\lambda_n(\mathcal{L}(\mathbf{B}))$, dove $\mathbf{B}^*$ è la base del reticolo generato da J^*. Inoltre $\det(\mathcal{L}(\mathbf{B_v})) < n^n \cdot ||f||^n / \det(\mathcal{L}(\mathbf{B}))$.*

Dimostrazione. Sia $\mathbf{w}$ un vettore di J^* di lunghezza $\lambda_1(\mathcal{L}(\mathbf{B}^*))$, generiamo un vettore $\mathbf{v} \in J^{-1}$ da $\mathbf{w} \in J^*$ come nel Lemma 3.18 e sia $\mathbf{B_v}$ la sua base di rotazione. Per il Lemma 3.18 si

ha $||\mathbf{B_v}|| \leq \sqrt{n} \cdot ||f|| \cdot ||\mathbf{w}||$ e per il fatto che $\lambda_1(\mathcal{L}(\mathbf{B})) \cdot \lambda_n(\mathcal{L}(\mathbf{B}^*)) \leq \sqrt{n}$ abbiamo $||\mathbf{w}|| \leq \sqrt{n}/\lambda_n(\mathcal{L}(\mathbf{B}))$, da cui discende la prima asserzione:

$$\lambda_n(\mathcal{L}(\mathbf{B_v})) \leq \sqrt{n} \cdot ||f|| \cdot \lambda_1(\mathcal{L}(\mathbf{B}^*)) \leq n \cdot ||f||/\lambda_n(\mathcal{L}(\mathbf{B})).$$

Mentre, per quanto riguarda la seconda affermazione, ricordiamo che $\det(\mathcal{L}(\mathbf{B}^*)) = 1/\det(\mathcal{L}(\mathbf{B}))$ e, dal Teorema di Minkowski, $||\mathbf{w}|| = \lambda_1(\mathcal{L}(\mathbf{B}^*)) < \sqrt{n}(\det(\mathcal{L}(\mathbf{B}^*)))^{1/n} = \sqrt{n}/(\det(\mathcal{L}(\mathbf{B})))^{1/n}$. Otteniamo allora $\det(\mathcal{L}(\mathbf{B_v})) < n^n \cdot ||f||^n/\det(\mathcal{L}(\mathbf{B}))$. □

Usiamo il Lemma appena enunciato per limitare $\lambda_n(\mathcal{L}(\mathbf{B}))$ in termini di n, $|f|$ e $\det(\mathcal{L}(\mathbf{B}))$.

Lemma 3.20. *Sia J un ideale di $R = \mathbb{Z}[x]/(f(x))$. Allora $\lambda_n(\mathcal{L}(\mathbf{B})) < n \cdot ||f|| \cdot \det(\mathcal{L}(B))^{1/n}$.*

Dimostrazione. Per il Lemma 3.19 abbiamo:

$$\lambda_n(\mathcal{L}(\mathbf{B})) \leq n \cdot ||f||/\lambda_n(\mathcal{L}(\mathbf{B_v})) \leq n \cdot ||f||/(\det(\mathcal{L}(\mathbf{B_v})))^{1/n} \leq n \cdot ||f|| \cdot (\det(\mathcal{L}(\mathbf{B})))^{1/n}. \quad \square$$

Avendo caratterizzato la relazione che intercorre tra l'inverso e il duale, possiamo ora illustrare le modifiche da apportare alla decodifica.

Modifica 1: Data una base $\mathbf{B}_I$ e la chiave segreta $\mathbf{B}_J^{\mathsf{sk}}$, calcoliamo un vettore $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$ e ridefiniamo la decodifica in modo che fornisca come output $\mathbf{m} = \mathbf{c} - \lfloor \mathbf{v}_J^{\mathsf{sk}} \times c \rceil \bmod \mathbf{B}_I$. Inoltre ridefiniamo $\mathcal{C}_{\mathcal{E}}$ in modo che usi $\mathcal{B}(r_{\mathsf{Dec}}/(n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||))$ invece di $\mathcal{B}(r_{\mathsf{Dec}})$.
Scopo: Semplificare l'equazione di decodifica e migliorare l'efficienza computazionale.

Questa modifica non cambia la profondità dei circuiti che eseguono la moltiplicazione (i circuiti che svolgono il prodotto tra matrice e vettore e la moltiplicazione nell'anello hanno la stessa profondità), tuttavia la Modifica 1 riduce la dimensione della chiave segreta e in tal modo contribuisce a diminuire la complessità della decodifica.
Per capire meglio questa semplificazione, supponiamo che $\mathbf{B}_J^{\mathsf{sk}}$ sia la base di rotazione per qualche vettore $\mathbf{w}_J^{\mathsf{sk}} \in \mathbb{Z}[x]/(f(x))$ e sia $\mathbf{x}_J^{\mathsf{sk}} = 1/\mathbf{w}_J^{\mathsf{sk}} \in \mathbb{Q}[x]/(f(x))$. La base di rotazione di $\mathbf{x}_J^{\mathsf{sk}}$ è proprio $(\mathbf{B}_J^{\mathsf{sk}})^{-1}$, pertanto

$$\mathbf{m} = (\mathbf{c} \bmod \mathbf{B}_J^{\mathsf{sk}}) \bmod \mathbf{B}_I = \mathbf{c} - \mathbf{B}_J^{\mathsf{sk}} \cdot \lfloor (\mathbf{B}_J^{\mathsf{sk}})^{-1} \cdot \mathbf{c} \rceil \bmod \mathbf{B}_I = \mathbf{c} - \mathbf{w}_J^{\mathsf{sk}} \times \lfloor \mathbf{x}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \bmod \mathbf{B}_I.$$

Abbiamo posto che $\mathbf{B}_J^{\mathsf{sk}}$ sia una base di rotazione, ma Gentry ha mostrato che la modifica funziona anche per basi che non verificano tale proprietà (Lemma 8.3.1 [12]):

Lemma 3.21. *Sia* $\mathbf{B}_J^{\mathsf{sk}}$ *una base iniziale segreta che decodifica correttamente per un dato parametro* r_{Dec}*. Da* $\mathbf{B}_J^{\mathsf{sk}}$ *e* $\mathbf{B}_I$ *possiamo calcolare in un tempo polinomiale un vettore* $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$ *tale che la base di rotazione di* $1/\mathbf{v}_J^{\mathsf{sk}}$ *circoscriva una palla di raggio almeno* $r_{\mathsf{Dec}}/(n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||)$*. In particolare, se* $\mathbf{c}$ *è un testo cifrato valido, nel senso che* $\mathbf{c} = \mathbf{m} + \mathbf{i} + \mathbf{j}$ *per un messaggio in chiaro* $\mathbf{m}$*,* $\mathbf{i} \in I$*,* $\mathbf{j} \in J$ *e* $\mathbf{m} + \mathbf{i} \in \mathcal{B}(r_{\mathsf{Dec}}/(n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||))$*, allora* $\mathbf{m} = \mathbf{c} - (\mathbf{v}_J^{\mathsf{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \bmod \mathbf{B}_I$*. Si può ulteriormente semplificare la decifratura, scegliendo in modo opportuno il valore di* $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$*, in tal caso si ha* $\mathbf{m} = \mathbf{c} - \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \bmod \mathbf{B}_I$*.*

Per poter semplificare il termine $(\mathbf{v}_J^{\mathsf{sk}})^{-1}$, Gentry richiede che $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$ sia anche contenuto in $1 + J^{-1}I$. Dato che I e J sono coprimi, esiste un vettore $\mathbf{j} \in J \cap (1 + I)$. Sia $\mathbf{r} = \mathbf{j} \times \mathbf{v}_J^{\mathsf{sk}}$ che è in R perché $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$. Inoltre per $\mathbf{v}_J^{\mathsf{sk}} \in 1 + J^{-1}I$ il vettore $\mathbf{r}$ appartiene a $1 + I$. Dai presupposti per la correttezza della decodifica (nel valore di r_{Dec} ridefinito sopra), noi sappiamo che $(\mathbf{v}_J^{\mathsf{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \in R$ e quindi risulta

$$(\mathbf{v}_J^{\mathsf{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil = \mathbf{r} \times (\mathbf{v}_J^{\mathsf{sk}})^{-1} \times \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \bmod I = \mathbf{j} \times \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil \bmod I = \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil,$$

completando così la semplificazione.

Modifica 2: Ridefiniamo l'insieme dei circuiti permessi $\mathcal{C}_{\mathcal{E}}$ sostituendo $\mathcal{B}(r_{\mathsf{Dec}})$ con $\mathcal{B}(r_{\mathsf{Dec}}/2)$. **Scopo**: Assicurarsi che i vettori che rappresentano i testi cifrati risultino più vicini ancora al reticolo J, in modo da semplificare la procedura di decodifica.

Dopo questa modifica il valore di r_{Dec} è la metà di quello utilizzato nel passo precedente e pertanto, rispetto al valore definito all'inizio, risulta

$$r'_{\mathsf{Dec}} = \frac{r_{\mathsf{Dec}}}{2n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||}.$$

Il fine sarà più chiaro una volta approfondito il circuito di decodifica nei dettagli, ma, brevemente, ricordiamo che $\mathsf{Decrypt}$ calcola $\mathbf{B}_J^{\mathsf{sk1}} \cdot \lfloor (\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{c} \rceil$, dove abbiamo usato la Modifica 1 e pertanto $\mathbf{B}_J^{\mathsf{sk1}}$ è la matrice identica e $(\mathbf{B}_J^{\mathsf{sk2}})^{-1}$ è la base di rotazione di $\mathbf{v}_J^{\mathsf{sk2}}$. Se permettiamo che i coefficienti di $(\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{c}$ siano molto vicini alla metà di un valore intero, avremo bisogno di una precisione molto alta per assicurarci che l'approssimazione fornisca il valore corretto. D'altra parte grazie alla seconda modifica abbiamo il seguente lemma.

Lemma 3.22. *Se* $\mathbf{c}$ *è un testo cifrato valido, considerata la Modifica 2, allora ogni coefficiente di* $(\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{c}$ *dista al più* $1/4$ *da un intero.*

Dimostrazione. Osserviamo che il vettore $\mathbf{c}$, corrispondente ad un testo cifrato, appartiene a $\mathcal{B}(r_{\mathsf{Dec}}) + J$. Si può quindi scrivere $\mathbf{c} = \mathbf{x} + \mathbf{j}$ con $\mathbf{x} \in \mathcal{B}(r_{\mathsf{Dec}})$ e $\mathbf{j} \in J$.
Allora si ha $(\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{c} = (\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{x} + (\mathbf{B}_J^{\mathsf{sk2}})^{-1} \cdot \mathbf{j}$, in cui il primo termine ha modulo al massimo $1/4$ per il Lemma 3.14 e l'ultimo è un vettore a coefficienti interi. □

Quanto detto aiuta a semplificare il circuito di decodifica senza diminuire in modo sostanziale la capacità di valutare circuiti di $\mathsf{Evaluate}$; infatti la massima profondità dei circuiti che ora possono essere valutati è diventata $\log\log(r_{\mathsf{Dec}}/2) - \log\log(\gamma_{\mathsf{Mult}}(R) \cdot r_{\mathsf{Enc}})$, che risulta minore di quella in origine, ma solo di un fattore additivo costante.

Riassumendo, il crittosistema dopo le modifiche è il seguente:
Creazione delle chiavi: $\mathsf{KeyGen}(R, \mathbf{B}_I)$

- Input: un anello R, una base $\mathbf{B}_I$ di un ideale I di R.
- Esegue $\mathsf{IdealGen}(R, \mathbf{B}_I)$: $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}}) \leftarrow \mathsf{IdealGen}(R, \mathbf{B}_I)$, basi di un ideale J di R; calcola $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$ tale che $\mathcal{P}(\mathbf{B}_{\mathbf{rot}}((\mathbf{v}_J^{\mathsf{sk}})^{-1})) \supset \mathcal{B}(r_{\mathsf{Dec}}/(2n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||))$.
- Output: $(\mathsf{pk}, \mathsf{sk})$;
  chiave pubblica: $\mathsf{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{\mathsf{pk}}, \mathsf{Samp})$;
  chiave segreta: $\mathsf{sk} = \mathbf{v}_J^{\mathsf{sk}}$.

Codifica: $\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m})$

- Input: pk, $\mathbf{m} \in \mathcal{M}$.
- Esegue Samp: $\mathbf{c}' \leftarrow \mathsf{Samp}(\mathbf{m}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathsf{pk}})$, da cui $\mathbf{c}' = \mathbf{m} + \mathbf{i}$, con $\mathbf{i} \in I$.
- Output: $\mathbf{c} \leftarrow \mathbf{c}' \bmod \mathbf{B}_J^{\mathsf{pk}}$.

Applicazione circuiti: $\mathsf{Evaluate}(\mathsf{pk}, C, \mathbf{c}_1, ..., \mathbf{c}_n)$

- Input: pk, un circuito $C \in \mathcal{C}_{\mathcal{E}}$, i testi cifrati $\mathbf{c}_1, ..., \mathbf{c}_n$.
- Applica il circuito generalizzato $g(C)$ di C ai testi cifrati: $\mathbf{c} \leftarrow g(C)(\mathbf{c}_1, ..., \mathbf{c}_n) \bmod \mathbf{B}_J^{\mathsf{pk}}$.
- Output: un ulteriore testo cifrato $\mathbf{c}$.

Decodifica: $\mathsf{Decrypt}(\mathbf{c}, \mathsf{sk})$

- Input: $\mathbf{c} \in R \bmod \mathbf{B}_J^{\mathsf{pk}}$.
- Calcola $\mathbf{m} = (\mathbf{c} - \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil) \bmod \mathbf{B}_I$.
- Output $\mathbf{m} \in \mathcal{M}$.

3.7 "Compressione" del circuito di decodifica (Squashing the Decryption Circuit)

L'idea di fondo su cui si basa questa ulteriore trasformazione è di inserire qualche informazione sulla chiave segreta $\mathbf{v}_J^{\mathsf{sk}}$ all'interno della chiave pubblica; di fatto inseriremo un grande insieme di vettori $\{\mathbf{t}_i \ : \ i = 1, \ldots, s\}$ che nasconde un sottoinsieme di vettori che sommati danno $\mathbf{v}_J^{\mathsf{sk}}$. Per garantire la sicurezza del sistema, apparentemente indebolito da questa modifica, faremo riferimento al problema computazionalmente difficile denominato Sparse Subset Sum Problem (SSSP). Per fare ciò abbiamo bisogno di due ulteriori algoritmi: SplitKey e ExpandCT.

SplitKey

L'algoritmo SplitKey è introdotto proprio al fine di inserire qualche informazione sulla chiave segreta nella chiave pubblica ed è parte della procedura di generazione delle chiavi KeyGen. Esso prende in input la chiave pubblica e quella segreta fornite come descritto precedentemente e dà come risultato un insieme casuale di vettori $\tau = \{\mathbf{t}_1, \ldots, \mathbf{t}_r\} \in J^{-1}$ tali che un sottoinsieme segreto S di τ contenga vettori che sommati diano la chiave segreta originale:

$$\mathbf{v}_J^{\mathsf{sk}} = \sum_{i \in S} \mathbf{t}_i \bmod I.$$

La nuova chiave segreta è una matrice $\mathbf{SK} = [sk_{ij}]$ composta da soli 0 e 1 che corrispondono all'insieme S nel seguente modo:

$$sk_{ij} = \begin{cases} 1 & \text{se e solo se } j \text{ è l'}i\text{-esimo elemento di } S, \\ 0 & \text{altrimenti.} \end{cases}$$

Mentre τ è aggiunto alla chiave pubblica, che diventa $\mathsf{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{\mathsf{pk}}, \mathsf{Samp}, \tau)$.

ExpandCT

L'algoritmo ExpandCT serve invece per preparare il testo cifrato al nuovo circuito di decodifica, che risulta meno profondo rispetto all'iniziale costruzione. Esso calcola

$$\mathbf{x}_i = \mathbf{t}_i \times \mathbf{c} \bmod \mathbf{B}_I, \qquad i = 1, \ldots, r,$$

dove $\mathbf{c}$ è il testo cifrato risultante da Encrypt. L'output finale è $\psi = \{\mathbf{c}, \mathbf{x}_1, \ldots, \mathbf{x}_r\}$.

Avendo modificato la chiave segreta e il testo cifrato, anche l'algoritmo Decrypt va corretto, prende in input la matrice $\mathbf{SK}$ e l'output di ExpandCT ψ. Conoscendo $\mathbf{SK}$ si può risalire ai "giusti" $\mathbf{x}_i$ (quelli con $i \in S$) e la decodifica funziona con la seguente equazione:

$$\mathbf{m} = \mathbf{c} - \Big\lfloor \sum_{i \in S} \mathbf{x}_i \Big\rceil \bmod \mathbf{B}_I.$$

Mostriamo la correttezza di quanto appena affermato.

Dimostrazione. Per quanto riguarda l'estrazione degli $\mathbf{x}_i$ rilevanti, si calcola un insieme di vettori $\{\mathbf{w}_{ij}\}$ con $i = 1, \ldots, |S|$ e $j = 1, \ldots, r$, $\mathbf{w}_{ij} = sk_{ij} \cdot \mathbf{x}_j$. Il vettore $\mathbf{w}_{ij}$ risulta diverso dal vettore nullo $\mathbf{0}$ se e solo se $sk_{ij} \neq 0$, facendo capire quali sono gli indici appartenenti ad S e quali sono i corrispondenti $\mathbf{x}_i$ di cui abbiamo bisogno.
Ora ricordiamo l'equazione di decodifica definita in precedenza e controlliamo che risulti coerente con quanto definito per il nuovo algoritmo Decrypt:

$$\begin{aligned} \mathbf{m} &= (\mathbf{c} - \lfloor \mathbf{v}_J^{\mathsf{sk}} \times \mathbf{c} \rceil) \bmod \mathbf{B}_I = \mathbf{c} - \Big\lfloor \Big(\sum_{i \in S} \mathbf{t}_i \Big) \times \mathbf{c} \Big\rceil \bmod \mathbf{B}_I \\ &= \mathbf{c} - \Big\lfloor \sum_{i \in S} \mathbf{t}_i \times \mathbf{c} \Big\rceil \bmod \mathbf{B}_I = \mathbf{c} - \Big\lfloor \sum_{i \in S} \mathbf{x}_i \Big\rceil \bmod \mathbf{B}_I. \end{aligned}$$

Quindi la modifica effettuata non inficia la correttezza del crittosistema. □

Mettiamo in evidenza quanto svolto finora per avere sempre presente il crittosistema $\mathcal{E}$ nella sua interezza:

Creazione delle chiavi: $\mathsf{KeyGen}(R, \mathbf{B}_I)$

- Input: un anello R, una base $\mathbf{B}_I$ di un ideale I di R.
- Esegue $\mathsf{IdealGen}(R, \mathbf{B}_I)$: $(\mathbf{B}_J^{\mathsf{sk}}, \mathbf{B}_J^{\mathsf{pk}}) \leftarrow \mathsf{IdealGen}(R, \mathbf{B}_I)$, basi di un ideale J di R;
  calcola $\mathbf{v}_J^{\mathsf{sk}} \in J^{-1}$ tale che $\mathcal{P}(\mathbf{B}_{\mathbf{rot}}((\mathbf{v}_J^{\mathsf{sk}})^{-1})) \supset \mathcal{B}(r_{\mathsf{Dec}}/(2n^{2.5} \cdot ||f|| \cdot ||\mathbf{B}_I||))$;
  fornisce $(\mathsf{pk}^*, \mathsf{sk}^*)$ tali che $\mathsf{pk}^* = (R, \mathbf{B}_I, \mathbf{B}_J^{\mathsf{pk}}, \mathsf{Samp})$ e $\mathsf{sk}^* = \mathbf{v}_J^{\mathsf{sk}}$;
  esegue $\mathsf{SplitKey}(\mathsf{pk}^*, \mathsf{sk}^*)$: $\tau = \{\mathbf{t}_1, \ldots, \mathbf{t}_r\}$, $S \subseteq \{1, \ldots, r\}$ tale che $\sum_{i \in S} \mathbf{t}_i \in \mathbf{v}_J^{\mathsf{sk}} + I$.
- Output:$(\mathsf{pk}, \mathsf{sk})$;
  chiave pubblica: $\mathsf{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{\mathsf{pk}}, \mathsf{Samp}, \tau)$;

chiave segreta: $\mathsf{sk} = \mathbf{SK} = [sk_{ij}]$ con $sk_{ij} = 1$ se e solo se j è l'i-esimo elemento di S o $sk_{ij} = 0$ altrimenti.

Codifica: $\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m})$

- Input: pk, $\mathbf{m} \in \mathcal{M}$.
- Esegue Samp: $\mathbf{c}' \leftarrow \mathsf{Samp}(\mathbf{m}, \mathbf{B}_I, R, \mathbf{B}_J^{\mathsf{pk}})$, da cui $\mathbf{c}' = \mathbf{m} + \mathbf{i}$, con $\mathbf{i} \in I$;
  calcola $\mathbf{c} \leftarrow \mathbf{c}' \bmod \mathbf{B}_J^{\mathsf{pk}}$;
  esegue $\mathsf{ExpandCT}(\mathbf{c}, \mathsf{pk})$: $\mathbf{x}_i = \mathbf{t}_i \times \mathbf{c} \bmod \mathbf{B}_I$ per $i \in \{1, \ldots, r\}$.
- Output: $\psi = (\mathbf{c}, \mathbf{x}_1, \ldots, \mathbf{x}_r)$.

Applicazione circuiti: $\mathsf{Evaluate}(\mathsf{pk}, C, \psi_1, ..., \psi_n)$

- Input: pk, un circuito $C \in \mathcal{C}_\mathcal{E}$, $\psi_1, \ldots, \psi_n$ risultanti da $\mathsf{Encrypt}$.
- Estrae $\mathbf{c}_1, \ldots, \mathbf{c}_n$ da $\psi_1, \ldots, \psi_n$;
  applica il circuito generalizzato $g(C)$ di C ai testi cifrati: $\mathbf{c} \leftarrow g(C)(\mathbf{c}_1, ..., \mathbf{c}_n) \bmod \mathbf{B}_J^{\mathsf{pk}}$;
  fornisce un ulteriore testo cifrato $\mathbf{c}$;
  applica $\mathsf{ExpandCT}(\mathbf{c}, \mathsf{pk})$ a $\mathbf{c}$ e si ottengono nuovi $\mathbf{x}_i$, $i = 1, \ldots, r$.
- Output: $\psi = (\mathbf{c}, \mathbf{x}_1, \ldots, \mathbf{x}_r)$.

Decodifica: $\mathsf{Decrypt}(\psi, \mathsf{sk})$

- Input: ψ fornito da $\mathsf{Evaluate}$.
- Trova $\mathbf{w}_{ij} = sk_{ij} \cdot \mathbf{x}_j \Rightarrow \{\mathbf{w}_{ij}\}$ con $\mathbf{w}_{ij} \neq \mathbf{0}$ sono gli elementi $\mathbf{x}_i$ con $i \in S$;
  calcola $\mathbf{m} = \left(\mathbf{c} - \left\lfloor \sum_{i \in S} \mathbf{x}_i \right\rceil\right) \bmod \mathbf{B}_I$.
- Output $\mathbf{m} \in \mathcal{M}$.

Questo crittosistema è in grado di valutare il proprio circuito di decodifica ($D_\mathcal{E} \subseteq \mathcal{C}_\mathcal{E}$) e, pertanto, è bootstrappable. Per mostrare tale risultato abbiamo bisogno prima del seguente lemma:

Lemma 3.23. *Per $i \in \{1, \ldots, t\}$, sia $a_i = (\ldots, a_{i,1}, a_{i,0}, a_{i,-1}, \ldots)$ un numero reale dato in rappresentazione binaria* mod $\mathbf{B}_I$ *tale che $\sum_i a_i$ disti dall'intero più vicino al più $1/4$. Esiste un circuito C (con le porte di addizione e moltiplicazione* mod $\mathbf{B}_I$*) che genera $t + 1$ interi*

$z_1, \ldots, z_{t+1}$ *(sempre in rappresentazione binaria), la cui somma è* $\lfloor \sum_i a_i \rceil$ *e tale che, se gli input del circuito generalizzato* $g(C)$ *sono in* $\mathcal{B}(r_{in})$*, allora i suoi output sono in* $\mathcal{B}(r_{out})$ *per:*

$$r_{out} \leq \left(\gamma(R) \cdot n \cdot ||\mathbf{B}_I|| \cdot (1 + \gamma(R) \cdot r_{in})^t \cdot t\right)^{\text{polylog}(t)}.$$

Per $||\mathbf{B}_I|| \leq r_{in}$ *e* $\gamma(R) = n^{\Omega(1)}$*, abbiamo:*

$$r_{out} \leq (\gamma(R) \cdot r_{in})^{t \cdot \text{polylog}(t)}.$$

Dimostrazione. Siano a_i^* la parte intera di a_i e $a_i^\dagger = (a_{i,-1}, a_{i,-2}, \ldots)$ la sua parte frazionaria, poniamo poi $T = \lceil \log t \rceil + 2$ e $b_i = (a_{i,-1}, a_{i,-2}, \ldots, a_{i,-T})$, ovvero tronchiamo la parte frazionaria $a_i^\dagger$ al T-esimo bit meno significativo. Asseriamo che $\lfloor \sum_i a_i^\dagger \rceil = \lfloor \sum_i b_i \rceil$, questo segue dal fatto che gli a_i sono presi in modo che la somma disti al massimo $1/4$ da un intero:

$$\left| \sum_i a_i^\dagger - \sum_i b_i \right| = \left| \sum_i \left(\sum_{j=T+1}^{+\infty} 2^{-j} \cdot a_{i,-j} \right) \right| < 1/4,$$

quindi i $t+1$ interi forniti in output dal circuito possono essere $a_1^*, \ldots, a_t^*, \lfloor \sum_i b_i \rceil$. La strategia per calcolare $\lfloor \sum_i b_i \rceil$ consiste per prima cosa nel valutare la rappresentazione binaria c_j del peso di Hamming di $(b_{1,-j}, \ldots, b_{t,-j})$ per ogni $j \in \{1, \ldots, T\}$:

$$\begin{array}{cccc} b_{1,-1} & b_{1,-2} & \ldots & b_{1,-T} \\ b_{2,-1} & b_{2,-2} & \ldots & b_{2,-T} \\ \vdots & \vdots & & \vdots \\ b_{t,-1} & b_{t,-2} & \ldots & b_{t,-T} \\ \hline c_1 & c_2 & \ldots & c_T \end{array}$$

Poi si procede col calcolare $\lfloor \sum_{j=1}^{T} 2^{-j} \cdot c_j \rceil$, che è computazionalmente più facile da ottenere della somma iniziale, in quanto consiste di soli T termini piuttosto che t.
La strategia diventa ancora più chiara quando $I = (2 \cdot \mathbf{e}_1)$ e lo spazio dei messaggi in chiaro è $\{0, 1\} \bmod I$.
In generale la rappresentazione binaria del peso di Hamming di $(x_1, \ldots, x_t)$ è data da

$$\left(e_{2^{\lfloor \log t \rfloor}}(x_1, \ldots, x_t) \bmod 2, \ldots, e_{2^0}(x_1, \ldots, x_t) \bmod 2\right)$$

dove $e_i(x_1, \ldots, x_t)$ è l'i-esimo polinomio simmetrico elementare in $x_1, \ldots, x_t$, che può essere calcolato efficientemente come l'i-esimo coefficiente del polinomio $p(z) = \prod_{j=1}^{t}(z - x_j)$. Il prossimo passo sarà cercare di limitare $||e_{2^k}(\mathbf{x}_1, \ldots, \mathbf{x}(t))||$ per $\mathbf{x}_i \in \mathcal{B}(r_{in})$ con $k \in \{0, \ldots, \lfloor \log t \rceil\}$.

Occorre precisare che per $I \neq (2 \cdot \mathbf{e}_1)$ la situazione diventa più complicata per il fatto che la riduzione modulo 2 non verifica automaticamente la riduzione modulo $\mathbf{B}_I$. Mostriamo allora un approccio differente che funziona anche nel caso più semplice di $I = (2 \cdot e_1)$. Sia $\mathbf{M} = [m_{ij}] \in M_{(t+1)\times(t+1)}(\mathbb{Z})$ data da $m_{ij} = \binom{i}{j}$ per $i, j \in \{0, \ldots, t\}$. Sia $\mathbf{M}^{-1}$ la matrice con elementi in $R \bmod I$ tale che $\mathbf{M}^{-1} \cdot \mathbf{M}$ sia la matrice identica modulo I, $\det \mathbf{M} = 1$ e $\mathbf{M}$ risulta invertibile modulo I. Prima di tutto il circuito calcola $\mathbf{v} \leftarrow \big(e_0(b_1, \ldots, b_t), \ldots, e_t(b_1, \ldots, b_t)\big)^T$. Notiamo che $\mathbf{M}^{-1} \cdot \mathbf{v} = \mathbf{e}_h$ che è essenzialmente il peso di Hamming h di $(b_1, \ldots, b_t)$ nell'unario (una lista di h valori uguali a 1 seguiti da uno 0). Dall'unario otteniamo la rappresentazione binaria del prodotto interno di $\mathbf{e}_h$ con $(\mathbf{c}_0, \ldots, \mathbf{c}_h, \ldots, \mathbf{c}_t)$, dove $\mathbf{c}_i$ è la rappresentazione binaria di i. Sia C il sotto-circuito mod $\mathbf{B}_I$ che calcola i bit della rappresentazione binaria del peso di Hamming. Usando $n \cdot ||\mathbf{B}_I||$ come limite superiore alla lunghezza degli elementi in $R \bmod \mathbf{B}_I$ abbiamo

$$\begin{aligned} ||g(C)(\mathbf{x}_1, \ldots, \mathbf{x}_t)|| &\leq \gamma(R) \cdot n \cdot ||\mathbf{B}_I|| \cdot \Big(\sum_{i=0}^{t} ||\mathbf{e}_i(\mathbf{x}_1, \ldots, \mathbf{x}_t)|| \Big) \cdot t \\ &\leq \gamma(R) \cdot n \cdot ||\mathbf{B}_I|| \cdot \Big(\sum_{i=0}^{t} \binom{i}{j} \gamma(R)^{i-1} \cdot r_{in}^{i} \Big) \cdot t \\ &= n \cdot ||\mathbf{B}_I|| \cdot (1 + \gamma(R) \cdot r_{in})^t \cdot t. \end{aligned}$$

A questo punto abbiamo generato T numeri, ognuno con $O(T)$ bit, che sommati danno $\sum_i b_i$. Esiste un circuito booleano di profondità $O(\log T)$ in grado di calcolare questa somma, che può essere emulato da un circuito mod $\mathbf{B}_I$ di profondità $O(\log T)$. Combinando quanto detto con il Teorema 3.9 otteniamo il risultato voluto. □

Teorema 3.24. *Il crittosistema $\mathcal{E}$ è bootstrappable quando*

$$|S| \cdot \log^{c_1} |S| \leq \frac{\log(r_{\mathsf{Dec}}/2)}{2^{c_2} \cdot \log(\gamma(R) \cdot r_{\mathsf{Enc}})},$$

dove $|S| \cdot \log^{c_1} |S|$ è il termine quasi-polinomiale che compare nel Lemma 3.23, $r_{\mathsf{Dec}}/2$ nasce dalla Modifica 2 e c_2 è una costante che rappresenta la profondità in un circuito avente le porte additive con $\gamma(R) = n^{\Omega(1)}$ fan-in e le porte moltiplicative con un numero costante di fan-in che esegua l'algoritmo di decodifica compresso.

Dimostrazione. Analogamente a quanto fatto nella dimostrazione del Teorema 3.9, per un circuito di livello c, se gli input del circuito generalizzato sono in $\mathcal{B}(r)$, gli output sono in

$\mathcal{B}((\gamma(R)\cdot r)^{2^c})$. Combinando questo con il Lemma 3.23, abbiamo che, se gli input del circuito di decodifica sono in $\mathcal{B}(r_{\mathsf{Enc}})$, allora l'output è in

$$(\gamma(R)\cdot r_{\mathsf{Enc}})^{2^{c_2}(|S|\cdot \mathrm{polylog}(|S|))}.$$

Si ottiene il risultato se il valore è al massimo $r_{\mathsf{Dec}}/2$:

$$(\gamma(R)\cdot r_{\mathsf{Enc}})^{2^{c_2}(|S|\cdot \mathrm{polylog}(|S|))} \leq r_{\mathsf{Dec}}/2$$

da cui, passando al logaritmo, si ha

$$2^{c_2}\cdot |S|\cdot \mathrm{polylog}(|S|)\log(\gamma(R)\cdot r_{\mathsf{Enc}}) \leq \log(r_{\mathsf{Dec}}/2),$$

ossia quanto voluto. □

Se, per esempio, supponiamo che $\gamma(R)\cdot r_{\mathsf{Enc}}$ sia polinomiale in n e che $r_{\mathsf{Dec}} = 2^{n^C}$ per $C < 1$, allora $|S|$ può essere sublineare in n.

3.8 Da bootstrappable a pienamente omomorfo

Dalla costruzione di un crittosistema bootstrappable, ovvero che annovera il proprio circuito di decodifica (o, meglio, sue versioni di poco aumentate) tra i circuiti permessi, otteniamo subito un crittosistema in grado di valutare circuiti di arbitraria profondità. Ma perché la bootstrappabilità è una caratteristica così potente? La ragione risiede nel fatto che essa permette di arginare l'aumento dell'errore dovuto alle varie operazioni contenute in Evaluate; un'operazione di "refresh" fatta sui testi cifrati. Per fare ciò un testo cifrato c, corrispondente ad un messaggio in chiaro m codificato tramite la chiave pubblica pk_i, viene ricodificato attraverso una chiave pubblica pk_{i+1} e al risultato viene poi applicato il circuito di decodifica in modo omomorfo, usando la codifica della chiave segreta sk_i sempre tramite la chiave pubblica pk_{i+1}; si ottiene così una cifratura di m tramite pk_{i+1} . Aver applicato l'algoritmo di decodifica ha pulito i testi cifrati dall'errore senza però rivelare mai il messaggio in chiaro originale, che risulta sempre coperto da almeno uno strato di codifica. Una volta che il messaggio è stato "pulito", si può proseguire con le operazioni omomorfe, mantenendo la correttezza del crittosistema.

Esempio. Immaginiamo una porta Add al livello $i+1$, prende in input la chiave segreta sk_i codificata tramite pk_{i+1}, che indichiamo nella figura con $\overline{^{i+1}\mathsf{sk}_i}$, e un insieme di testi cifrati legati

al livello i che sono codificati tramite pk_i. La procedura valuta in modo omomorfo il circuito di decodifica aumentato di una porta additiva, contrassegnato con D_{Add}, come mostrato in Figura 3.2.

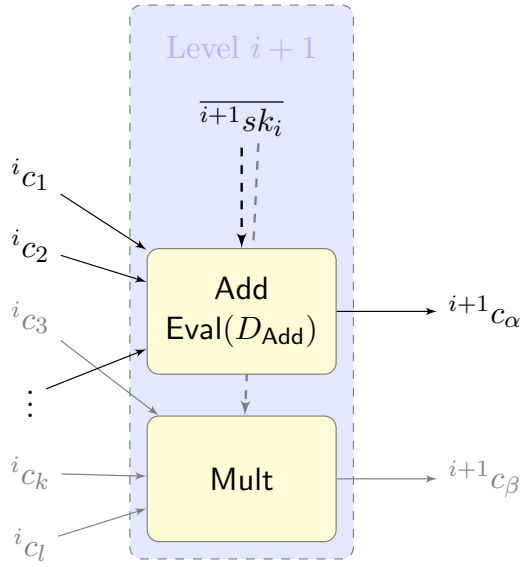

Figura 3.2: Una porta Add al livello $i+1$

Per vedere come funziona nel dettaglio, iniziamo considerando l'algoritmo chiamato Recrypt. Per semplicità supponiamo che lo spazio dei messaggi in chiaro $\mathcal{M}$ sia $\{0,1\}$ e il circuito di decodifica, aumentato di qualche porta, $D_{\mathcal{E}}$ sia un circuito booleano in $\mathcal{C}_{\mathcal{E}}$. Siano $(\mathsf{sk}_1, \mathsf{pk}_1)$ e $(\mathsf{sk}_2, \mathsf{pk}_2)$ due coppie di chiavi per $\mathcal{E}$, denotiamo con c_1 la codifica del messaggio m tramite pk_1 e con $\overline{\mathsf{sk}_{1j}}$ la codifica del j-esimo bit della prima chiave segreta sk_1 tramite la seconda chiave pubblica pk_2 . L'algoritmo allora è il seguente:

$\underline{\mathsf{Recrypt}(\mathsf{pk}_2, D_{\mathcal{E}}, \langle\overline{\mathsf{sk}_{1j}}\rangle, c_1)}$

- Input: pk_2, il circuito $D_{\mathcal{E}}$, i bit codificati della prima chiave segreta $\langle\overline{\mathsf{sk}_{1j}}\rangle$ e il testo cifrato c_1.
- Pone $\overline{c_{1j}} \leftarrow \mathsf{Encrypt}(\mathsf{pk}_2, c_{1j})$, dove c_{1j} è il j-esimo bit di c_1;
  calcola $c_2 \leftarrow \mathsf{Evaluate}\big(\mathsf{pk}_2, D_{\mathcal{E}}, \langle\langle\overline{\mathsf{sk}_{1j}}\rangle, \langle\overline{c_{1j}}\rangle\rangle\big)$.
- Output: c_2 risulta una codifica tramite pk_2 di $\mathsf{Decrypt}(\mathsf{sk}_1, c_1) \to m$.

Definizione 3.25. *Sia $D_{\mathcal{E}}$ il circuito di decodifica di un crittosistema $\mathcal{E}$ che prende in input una chiave segreta e un testo cifrato, ognuno della forma di un elemento di $\mathcal{M}^{l(\lambda)}$, dove $\mathcal{M}$ è lo spazio dei messaggi in chiaro e l denota la lunghezza di bit della chiave segreta. Sia Γ un insieme di porte con input e output in $\mathcal{M}$, che include le porte banali (per le quali gli input e gli output sono gli stessi). Un circuito composto da copie di $D_{\mathcal{E}}$ connesse da una singola porta* g *(il numero di copie deve essere uguale al numero di input di* g*) è chiamato* ***circuito di decodifica*** g***-aumentato****. L'insieme di tali circuiti con* $\mathsf{g} \in \Gamma$ *è denominato* $D_{\mathcal{E}}(\Gamma)$.

Con questa nozione di circuito di decodifica aumentato, la condizione di bootstrappabilità diventa:

Definizione 3.26. *Sia $\mathcal{C}_{\mathcal{E}}$ l'insieme dei circuiti permessi che il crittosistema $\mathcal{E}$ è in grado di valutare in modo compatto, allora $\mathcal{E}$ è bootstrappable rispetto a Γ se $D_{\mathcal{E}}(\Gamma) \subseteq \mathcal{C}_{\mathcal{E}}$.*

Per esempio, se Γ include la porta banale e NAND, $\mathcal{E}$ è bootstrappable rispetto a Γ se $\mathcal{C}_{\mathcal{E}}$ contiene $D_{\mathcal{E}}$ e il circuito formato da due copie di $D_{\mathcal{E}}$ collegate mediante una porta NAND.
A partire da un crittosistema in grado di valutare in modo compatto questi due circuiti possiamo ottenere un crittosistema pienamente omomorfo livellato, che definiamo qui di seguito.

Definizione 3.27. *Diciamo che una famiglia di crittosistemi omomorfi $\{\mathcal{E}^{(d)} : d \in \mathbb{Z}^+\}$ è* ***pienamente omomorfa livellata*** *(leveled fully homomorphic) se, per ogni $d \in \mathbb{Z}^+$, i crittosistemi usano lo stesso circuito di decodifica e $\mathcal{E}^{(d)}$ valuta in modo compatto tutti i circuiti di profondità d (con uno specificato insieme di porte Γ) e la complessità computazionale degli algoritmi di $\mathcal{E}^{(d)}$ è polinomiale in λ, d e, nel caso di* Evaluate*, nella dimensione del circuito C.*

In dettaglio mostriamo come uno schema bootstrappable possa essere usato per costruire un crittosistema pienamente omomorfo livellato. Sia $\mathcal{E}$ bootstrappable rispetto ad un insieme di porte Γ e, per ogni $d \geq 1$, usiamo $\mathcal{E}$ per costruire uno schema

$$\mathcal{E}^{(d)} = (\mathsf{KeyGen}_{\mathcal{E}^{(d)}}, \mathsf{Encrypt}_{\mathcal{E}^{(d)}}, \mathsf{Evaluate}_{\mathcal{E}^{(d)}}, \mathsf{Decrypt}_{\mathcal{E}^{(d)}})$$

che valuti tutti i circuiti di profondità d aventi le porte in Γ. Osserviamo che, nella descrizione che seguirà, è stato scelto di cifrare la chiave segreta in ordine inverso per semplificare la descrizione di Evaluate.
Usiamo la notazione $D_{\mathcal{E}}(\Gamma, \delta)$ per indicare l'insieme dei circuiti che eguagliano un circuito di profondità δ con le porte in Γ i cui input sono copie di $D_{\mathcal{E}}$.

Creazione delle chiavi: $\mathsf{KeyGen}_{\mathcal{E}^{(d)}}(\lambda, d)$

- Input: un parametro di sicurezza λ e un intero positivo d.
- $(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\mathcal{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, per $i \in \{0, \ldots, d\}$;
  $\overline{\mathsf{sk}_{ij}} \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}_{i-1}, \mathsf{sk}_{ij})$ per $i \in \{1, \ldots, d\}$, $j \in \{1, \ldots, l\}$ (l è la lunghezza di bit della chiave segreta e sk_{ij} è la rappresentazione binaria di sk_i).
- Output: la chiave segreta $\mathsf{sk}^{(d)} \leftarrow \mathsf{sk}_0$ e la chiave pubblica $\mathsf{pk}^{(d)} \leftarrow \left(\langle \mathsf{pk}_i \rangle, \langle \overline{\mathsf{sk}_{ij}} \rangle\right)$;
  denotiamo con $\mathcal{E}^{(\delta)}$ il sottosistema che usa $\mathsf{sk}^{(\delta)} \leftarrow \mathsf{sk}_0$ e $\mathsf{pk}^{(\delta)} \leftarrow \left(\langle \mathsf{pk}_i \rangle, \langle \overline{\mathsf{sk}_{ij}} \rangle\right)$ con $1 \leq i \leq \delta \leq d$.

Codifica: $\mathsf{Encrypt}_{\mathcal{E}^{(d)}}(\mathsf{pk}^{(d)}, m)$

- Input: $\mathsf{pk}^{(d)}$, $m \in \mathcal{M}$.
- Output: il testo cifrato $c \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}_d, m)$.

Applicazione circuiti: $\mathsf{Evaluate}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C_\delta, \Psi_\delta)$

- Input: $\mathsf{pk}^{(\delta)}$, un circuito C_δ di profondità al massimo δ con porte in Γ, un insieme $\mathbf{\Psi}_\delta$ di testi cifrati tramite pk_δ.
- Assumiamo che ogni arco di C_δ connetta porte di livello consecutivo, se così non fosse aggiungiamo una porta banale per realizzarlo.
  Se $\delta = 0$ fornisce in output Ψ_0 e termina.
  Altrimenti:
  - pone $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1}) \leftarrow \mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C_\delta, \Psi_\delta)$;
  - pone $(C_{\delta-1}, \Psi_{\delta-1}) \leftarrow \mathsf{Reduce}_{\mathcal{E}^{(\delta-1)}}(\mathsf{pk}^{(\delta-1)}, C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$;
  - esegue $\mathsf{Evaluate}_{\mathcal{E}^{(\delta-1)}}(\mathsf{pk}^{(\delta-1)}, C_{\delta-1}, \Psi_{\delta-1})$.
- Output: $c \leftarrow \Psi_0$, un testo cifrato tramite pk_0.

Decodifica: $\mathsf{Decrypt}_{\mathcal{E}^{(d)}}(\mathsf{sk}^{(d)}, c)$

- Input: $\mathsf{sk}^{(d)}$ e c, un testo cifrato (tramite pk_0).
- Output: $m \leftarrow \mathsf{Decrypt}_{\mathcal{E}}(\mathsf{sk}_0, c)$.

All'interno di Evaluate abbiamo usato gli algoritmi Augment e Reduce che descriviamo a seguire.

$\underline{\mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C_\delta, \mathbf{\Psi}_\delta)}$

- Input: $\mathsf{pk}^{(\delta)}$, un circuito C_δ di profondità al massimo δ con porte in Γ, un insieme $\mathbf{\Psi}_\delta$ di testi cifrati tramite pk_δ.
- Aumenta C_δ con $D_\mathcal{E}$ e chiama il circuito risultante $C^\dagger_{\delta-1}$.
  Sia $\Psi^\dagger_{\delta-1}$ l'insieme dei testi cifrati formati sostituendo ogni testo cifrato in input $c \in \Psi_\delta$ con $\langle\langle\overline{\mathsf{sk}_{\delta j}}\rangle, \langle\overline{c_j}\rangle\rangle$, dove $\overline{c_j} \leftarrow \mathsf{Encrypt}_{\mathcal{E}^{\delta-1}}(\mathsf{pk}^{(\delta-1)}, c_j)$ e c_j è il j-esimo bit di c scritto in rappresentazione binaria.
- Output: $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$.

$\underline{\mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C^\dagger_\delta, \mathbf{\Psi}^\dagger_\delta)}$

- Input: $\mathsf{pk}^{(\delta)}$, un insieme di testi cifrati $\Psi^\dagger_\delta$ (immagini di $\mathsf{Encrypt}_{\mathcal{E}^\delta}$) e un circuito $C^\dagger_\delta \in D_\mathcal{E}(\Gamma, \delta+1)$.
- Pone C_δ uguale al sottocircuito di $C^\dagger_\delta$ che consiste nei primi δ livelli.
  Fornisce Ψ_δ, dato dai testi cifrati indotti come input al circuito C_δ, dove il testo cifrato $c^{(w)}_\delta$ associato all'arco w del livello δ è il risultato di $\mathsf{Evaluate}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C^{(w)}_\delta, \Psi^{(w)}_\delta)$, avendo posto $C^{(w)}_\delta$ il sottocircuito di $C^\dagger_\delta$ con arco di output w e $\Psi^{(w)}_\delta$ i testi cifrati input di $C^{(w)}_\delta$.
- Output: (C_δ, Ψ_δ).

Il sistema descritto risulta corretto, per mostrare questo abbiamo il seguente teorema.

Teorema 3.28. *Sia $\mathcal{E}$ bootstrappable rispetto a un insieme di porte Γ, allora $\mathcal{E}^{(d)}$ valuta in modo compatto tutti i circuiti di profondità d con porte in Γ, in altre parole $\mathcal{E}^{(d)}$ è pienamente omomorfo livellato.*

Dimostrazione. Sia $D(\delta, w, C, \Psi)$ il valore in chiaro per l'arco w in un circuito C indotto dalla decodifica svolta rispetto a sk_δ dei testi cifrati Ψ associati agli input di C. Se C è vuoto, cioé non ha porte, allora gli archi in entrata sono uguali a quelli in uscita e $D(\delta, w, C, \Psi)$ denota solo la decodifica del singolo testo cifrato $c \in \Psi$ associato a w.
Per provare la correttezza basta mostrare che

$$D(d, w_{out}, C_d, \Psi_d) = D(0, w_{out}, C_0, \Psi_0)$$

per ogni arco in output w_{out} di C_0 (il circuito C al livello 0).
Innanzitutto, quando $(C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1}) \leftarrow \mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C_\delta, \boldsymbol{\Psi}_\delta)$, possiamo affermare che

$$D(\delta, w, C_\delta, \Psi_\delta) = D(\delta - 1, w, C^\dagger_{\delta-1}, \Psi^\dagger_{\delta-1})$$

per ogni porta w al livello al massimo $\delta - 1$; questo segue dalla correttezza di $\mathsf{Recrypt}$ e dal fatto che i circuiti C_δ e $C^\dagger_{\delta-1}$ sono identici fino al livello $\delta - 1$.
Successivamente, quando $(C_\delta, \Psi_\delta) \leftarrow \mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C^\dagger_\delta, \boldsymbol{\Psi}^\dagger_\delta)$, abbiamo

$$D(\delta, w, C^\dagger_\delta, \Psi^\dagger_\delta) = D(\delta, w, C_\delta, \Psi_\delta)$$

per ogni arco w al livello al massimo δ; quanto appena affermato segue dalla correttezza di $\mathsf{Evaluate}_{\mathcal{E}}$ per i circuiti in $D_{\mathcal{E}}(\Gamma)$ e dal fatto che i circuiti $C^\dagger_\delta$ e C_δ sono uguali fino al livello δ.
Le due asserzioni forniscono il risultato desiderato: $D(d, w_{out}, C_d, \Psi_d) = D(0, w_{out}, C_0, \Psi_0)$. □

Notiamo che Γ è arbitrario; per esempio ogni porta in esso contenuta può essere a sua volta un circuito di $(\mathsf{AND}, \mathsf{OR}, \mathsf{NOT})$ di profondità m e fan-in 2, rendendo corretta la valutazione dei circuiti fino alla profondità $d \cdot m$ per il Teorema 3.28.

Teorema 3.29. *Per un circuito C di profondità al massimo d e dimensione s, definita qui come il numero di archi, la complessità computazionale dell'applicazione di* $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ *a C è dominata da al più $s \cdot l \cdot d$ applicazioni di* $\mathsf{Encrypt}_{\mathcal{E}}$ *e da al più $s \cdot d$ applicazioni di* $\mathsf{Evaluate}_{\mathcal{E}^{(d)}}$ *ai circuiti in $D_{\mathcal{E}}(\Gamma)$.*

Dimostrazione. Consideriamo una fase di pre-elaborazione per assicurarci che tutti gli archi nel circuito colleghino porte di livello consecutivo; questa fase incrementa il numero di archi nel circuito di un fattore moltiplicativo al massimo pari a d. Dobbiamo provare che la complessità computazionale, per questa fase, è dominata da al più $s' \cdot l$ applicazioni di $\mathsf{Encrypt}$ e al massimo s' applicazioni di $\mathsf{Evaluate}_{\mathcal{E}}$ ai circuiti in $D_{\mathcal{E}}(\Gamma)$, dove s' è la dimensione del circuito pre-elaborato.
La complessità di $\mathsf{Augment}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C_\delta, \Psi_\delta)$ è dominata dalla sostituzione di ogni testo cifrato $c \in \Psi_\delta$ con $\langle\langle \overline{\mathsf{sk}_{\delta j}} \rangle, \langle \overline{c_j} \rangle\rangle$; generando i $\langle \overline{c_j} \rangle$ si inducono $|W_\delta| \cdot l$ applicazioni di $\mathsf{Encrypt}_{\mathcal{E}}$, dove W_δ è l'insieme degli archi al livello δ. Sommando su δ otteniamo al massimo $s' \cdot l$ esecuzioni di $\mathsf{Encrypt}_{\mathcal{E}}$.
La complessità di $\mathsf{Reduce}_{\mathcal{E}^{(\delta)}}(\mathsf{pk}^{(\delta)}, C^\dagger_\delta, \Psi^\dagger_\delta)$ è dominata dalla valutazione di $C^{(w)}_\delta$ per ogni $w \in W_\delta$, che induce $|W_\delta|$ applicazioni di $\mathsf{Evaluate}_{\mathcal{E}}$ ai circuiti in $D_{\mathcal{E}}(\Gamma)$. Sommando rispetto a δ, abbiamo al massimo s' di tali applicazioni. Pertanto consegue che la complessità risulta quella enunciata. □

In conclusione, assumendo la sicurezza semantica di $\mathcal{E}$, proviamo la sicurezza di $\mathcal{E}^{(d)}$. Ricordiamo che, affinché un sistema crittografico si possa definire semanticamente sicuro, un attaccante che disponga di una potenza di calcolo limitata non deve essere in grado di derivare delle informazioni significative sul sistema scegliendo testi in chiaro arbitrari e osservando la generazione dei testi cifrati corrispondenti; in altre parole ci riferiamo alla (t, ε)-sicurezza semantica rispetto agli attacchi di tipo *chosen-plaintext* (CPA) e intendiamo così che non ci sono algoritmi conosciuti che violino il crittosistema in un tempo polinomiale t con un vantaggio ε non trascurabile nel parametro di sicurezza.
Per un ordinario cifrario asimmetrico la sicurezza è caratterizzata dalle seguenti fasi (che Gentry racchiude nel termine Gioco [12]):

- Configurazione (*setup*): lo sfidante genera in modo casuale le chiavi $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\mathcal{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda)$ (λ parametro di sicurezza) e dà pk all'avversario $\mathcal{A}$.

- Sfida (*challenge*): $\mathcal{A}$ genera due messaggi in chiaro $m_0^*, m_1^* \in \mathcal{M}$ e li manda allo sfidante, che fissa in modo aleatorio un valore di $b \xleftarrow{\mathcal{R}} \{0,1\}$ e codifica il messaggio in chiaro corrispondente $c^* \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}, m_b^*)$. Invia poi c^* ad $\mathcal{A}$.

- Congettura (*guess*): $\mathcal{A}$ manda $b' \in \{0,1\}$ allo sfidante e vince il gioco se $b' = b$.

L'unica differenza tra la sicurezza semantica per un qualsiasi cifrario asimmetrico e quella per un crittosistema omomorfo consiste nel fatto che, nel secondo caso, l'avversario può mandare più testi cifrati nella fase di sfida:

- Sfida (nel caso omomorfo): $\mathcal{A}$ manda allo sfidante un circuito $C \in \mathcal{C}_{\mathcal{E}}$ avente $k = \text{poly}(\lambda)$ input e due insiemi di messaggi in chiaro $(m_{01}, \ldots, m_{0k}), (m_{11}, \ldots, m_{1k}) \in \mathcal{M}^k$; lo sfidante pone $b \xleftarrow{\mathcal{R}} \{0,1\}$ e fornisce $c^* \xleftarrow{\mathcal{R}} \mathsf{Evaluate}_{\mathcal{E}}(\mathsf{pk}, C, c_{b1}, \ldots, c_{bk})$, dove $c_{bi} \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}, m_{bi})$. Si può semplificare la fase chiedendo all'avversario solo i testi cifrati in input $c_{b1}, \ldots, c_{bk}$, in quanto anche $\mathcal{A}$ può eseguire $\mathsf{Evaluate}_{\mathcal{E}}$.

Detto questo possiamo enunciare il teorema che dimostri la sicurezza di $\mathcal{E}^{(d)}$.

Teorema 3.30. *Sia $\mathcal{A}$ un algoritmo che violi la sicurezza semantica di $\mathcal{E}^{(d)}$ con un vantaggio pari a ε in un tempo t, allora esiste un algoritmo $\mathcal{B}$ che viola la sicurezza semantica di $\mathcal{E}$ per $t' \approx t \cdot l$ con un vantaggio $\varepsilon' \geq \varepsilon / l(d+1)$, per l la lunghezza di bit della chiave segreta.*

Dimostrazione. Sia $(\mathcal{E})^l$ il crittosistema equivalente a $\mathcal{E}$, ma avente come spazio dei messaggi in chiaro $\mathcal{M}^{\leq l}$ e per il quale $\mathsf{Encrypt}_{(\mathcal{E})^l}$ coinvolge fino a l invocazioni di $\mathcal{E}$ concatenando i risultati. Il nostro obiettivo consiste nel mostrare che $\mathcal{B}$ viola la sicurezza semantica di $(\mathcal{E})^l$ con parametri (t'', ε''), $t'' \approx t$ e $\varepsilon'' \geq \varepsilon/(d+1)$, da cui segue il risultato dell'enunciato.

Per $k \in [0, d]$, indichiamo con il Gioco(k) un attacco contro $\mathcal{E}^{(d)}$ che proceda analogamente all'algoritmo $\mathcal{A}$ eccetto che per ogni $i \in [1, k]$ colui che attacca codifica mediante la chiave pubblica pk_{i-1} il j-esimo bit di una chiave segreta sk'_i scelta casualmente e non legata a sk_i. Il Gioco($d+1$) è uguale al Gioco(d) tranne che per il fatto che l'attaccante, ignorando i messaggi in chiaro reali m_0, m_1 e b, genera un testo in chiaro m di appropriata lunghezza e lo codifica generando un testo cifrato di sfida. Sia ε_k il vantaggio dell'avversario nel Gioco(k).

Dato che il Gioco(0) risulta essere identico a quello dell'attacco $\mathcal{A}$, il vantaggio è ε, inoltre $\varepsilon_{d+1} = 0$, dato che la sfida è indipendente da b. Di conseguenza, per qualche $k \in [0, d]$, deve valere $|\varepsilon_k - \varepsilon_{k+1}| \geq \varepsilon/(d+1)$; fissiamo questo valore di k.

Descriviamo di seguito come $\mathcal{B}$ usi $\mathcal{A}$ per violare $(\mathcal{E})^l$. $\mathcal{B}$ riceve dallo sfidante una chiave pubblica pk e genera i valori pubblici e segreti come nel Gioco(k), ma sostituisce il suo valore di pk_k con la chiave pubblica pk fornitagli. Inoltre, se $k < d$, genera una finta coppia di chiavi $(\mathsf{sk}'_{k+1}, \mathsf{pk}'_{k+1}) \xleftarrow{\mathcal{R}} \mathsf{KeyGen}_{\mathcal{E}}(\lambda)$ e pone $m_0 \leftarrow \mathsf{sk}_{k+1}$ e $m_1 \leftarrow \mathsf{sk}'_{k+1}$; richiede poi un testo cifrato (mediante pk) di sfida che codifichi sia m_0 che m_1 appartenenti a $\mathcal{M}^l$. Lo sfidante genera $\beta \xleftarrow{\mathcal{R}} \{0,1\}$ e manda l'insieme di testi cifrati $\langle c_j \rangle$ che codificano $\langle m_{\beta j} \rangle$. $\mathcal{B}$ rimpiazza il suo insieme originale $\langle \overline{\mathsf{sk}_{(k+1)j}} \rangle$ con $\langle c_j \rangle$. Otteniamo che i valori della chiave pubblica sono generati come nel Gioco($k+\beta$) e $\mathcal{B}$ li invia ad $\mathcal{A}$.

Eventualmente, $\mathcal{A}$ può richiedere una codifica di m_0 o m_1. In tal caso $\mathcal{B}$ fissa $b \xleftarrow{\mathcal{R}} \{0,1\}$ e, se $k < d$, manda $c_j \xleftarrow{\mathcal{R}} \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}_d, m_{bj})$, mentre, se $k = d$, $\mathcal{B}$ genera $m \xleftarrow{\mathcal{R}} \mathcal{M}$ e chiede allo sfidante una codifica di m_b o m. Lo sfidante genera $\beta \xleftarrow{\mathcal{R}} \{0,1\}$ e codifica rispettivamente m_b o m. $\mathcal{B}$ inoltra il risultato ad $\mathcal{A}$, che a sua volta rimanda un bit b'. A questo punto $\mathcal{B}$ invia allo sfidante $\beta' \leftarrow b \oplus b'$. Risulta che la sfida è generata allo stesso modo del Gioco($k+\beta$).

Dato che la simulazione di $\mathcal{B}$ ha la stessa distribuzione del Gioco($k+\beta$), la probabilità che $\mathcal{B}$ dia in output 0 è $\varepsilon_{k+\beta}$ e il risultato segue. □

Capitolo 4

Un'implementazione del crittosistema pienamente omomorfo

Il primo tentativo di migliorare l'implementazione del crittosistema di Gentry è stato realizzato nel 2009 da N.P. Smart e F. Vercauteren [29], che lo hanno attuato usando un reticolo generato da un ideale principale con la condizione che abbia determinante primo. Purtroppo tale implementazione ha incontrato alcune difficoltà: la necessità di generare molti candidati per trovare il reticolo con determinante primo; l'alta complessità del calcolo della chiave segreta che gli corrisponde (ciò rende inaccettabili reticoli di dimensione alta, che però sono più sicuri); inoltre il polinomio corrispondente alla fase di decodifica compressa ha grado stimato troppo elevato per essere accettabile. L'implementazione che riportiamo prosegue la direzione di Smart e Vercauteren ed è dovuta allo stesso C. Gentry insieme a S. Halevi nel 2011 [15]. L'innovazione consiste nella creazione di un nuovo algoritmo più veloce per ottenere la chiave segreta e nell'eliminare la richiesta che il determinante del reticolo sia primo, nonché in alcune ottimizzazioni che portano ad avere un polinomio relativo alla decodifica di grado molto più basso (solamente quindici, rispetto ad alcune centinaia di prima). Possiamo così ottenere un crittosistema pienamente omomorfo e sicuro per una dimensione di reticolo $n = 2^{13}$ o $n = 2^{15}$. Il capitolo sarà diviso in due parti per mettere in risalto il passaggio da un sistema parzialmente omomorfo ad uno pienamente omomorfo.

4.1 Parte I: Crittosistema parzialmente omomorfo

4.1.1 Generazione delle chiavi

Come Smart e Vercauteren, anche Gentry e Halevi lavorano con un anello $R = \mathbb{Z}/(f_n(x))$, in cui $f_n(x) = x^n + 1$ e n è una potenza di due, scegliendo un ideale principale J di R a cui far corrispondere la nozione di reticolo; ma, invece di richiedere che il determinate di tale reticolo sia primo, pongono solo la condizione che la matrice composta dai vettori di base del reticolo sia in forma normale hermitiana come segue

$$\mathrm{HFN}(J) = \begin{bmatrix} d & -r & -[r^2]_d & -[r^3]_d & \dots & -[r^{n-1}]_d \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{4.1}$$

dove d è il determinante del reticolo e r è una radice di $f_n(x)$ modulo d. In particolare notiamo che il simbolo $[\cdot]_d$ indica la riduzione modulo d con risultato in $[-d/2, d/2)$.
La fase di generazione delle chiavi è composta dai seguenti passi:

1. Scegliamo un elemento $v(x)$ di R; esso sarà nella forma $v(x) = \sum_{i=0}^{n-1} v_i x^i$. Identifichiamo tale elemento con il vettore $\mathbf{v}$ formato dai coefficienti di $v(x)$ (che sono di lunghezza fissata pari a t bit). Associamo a $\mathbf{v}$ la base di rotazione $n \times n$:

$$\mathbf{B_v} = \begin{bmatrix} v_0 & -v_{n-1} & -v_{n-2} & \dots & -v_1 \\ v_1 & v_0 & -v_{n-1} & \dots & -v_2 \\ v_2 & v_1 & v_0 & \dots & -v_3 \\ & & & \ddots & \\ v_{n-1} & v_{n-2} & v_{n-3} & \dots & v_0 \end{bmatrix}, \tag{4.2}$$

   dove l'i-esima colonna è data dai coefficienti ottenuti calcolando $v_i(x) = v(x) \times x^i \bmod f_n(x)$.

2. Calcoliamo l'inverso di $v(x) \bmod f_n(x)$ a meno di un fattore moltiplicativo, ovvero un polinomio $w(x)$ di grado al massimo $n-1$ tale che $w(x) \times v(x) = d (\bmod f_n(x))$, dove d è una costante coincidente con il determinante del reticolo $\mathcal{L}(\mathbf{B_v})$, nonché con il risultante dei polinomi $v(x)$ e $f_n(x)$, dato che f_n è monico. Si ha che la matrice

$$\mathbf{B_w} = \begin{bmatrix} w_0 & -w_{n-1} & -w_{n-2} & \dots & -w_1 \\ w_1 & w_0 & -w_{n-1} & \dots & -w_2 \\ w_2 & w_1 & w_0 & \dots & -w_3 \\ & & & \ddots & \\ w_{n-1} & w_{n-2} & w_{n-3} & \dots & w_0 \end{bmatrix}$$

è tale per cui $\mathbf{B_w} \times \mathbf{B_v} = \mathbf{B_v} \times \mathbf{B_w} = d \cdot \mathbf{I}$, con $\mathbf{I}$ la matrice identica. Un modo per calcolare il polinomio $w(x)$ è applicando l'algoritmo euclideo del massimo comune divisore esteso (Extended Euclidean-GCD) a $v(x)$ e $f_n(x)$, ma ci sono metodi più efficienti (quali la trasformata di Fourier veloce, FFT).

3. Verifichiamo che $v(x)$ soddisfi la condizione che la forma hermitiana normale di $\mathbf{B_v}$ sia come in (4.1).

   Lemma 4.1. *La forma hermitiana normale di una matrice* $\mathbf{B_v}$*, definita in* (4.2)*, risulta nella forma* (4.1) *se e solo se il reticolo generato dalle righe di* $\mathbf{B_v}$ *contiene un vettore della forma* $\mathbf{r} = (-r, 1, 0, \ldots, 0)^T$.

   Dimostrazione. Sia $\mathbf{H}$ la forma hermitiana di $\mathbf{B_v}$, $\mathbf{H}$ è triangolare inferiore con elementi in diagonale non negativi, le cui righe generano lo stesso reticolo ottenuto dalle righe di $\mathbf{B_v}$ e ogni elemento al di sotto della diagonale non supera la metà dell'elemento in diagonale corrispondente. Questa matrice può essere ottenuta da $\mathbf{B_v}$ mediante una serie di operazioni elementari sulle righe ed è unica.
   Se $\mathbf{H}$ è nella forma descritta in (4.1) allora è chiaro che un vettore $\mathbf{r}$ esiste necessariamente e coincide con la trasposta della seconda riga della matrice. Proviamo che tale condizione è anche sufficiente.
   Il vettore $d \cdot \mathbf{e}_1 = (d, 0, \ldots, 0)^T$ appartiene sicuramente al reticolo $\mathcal{L}(\mathbf{B_v})$, in particolare sappiamo che $\mathbf{B_v} \times (w_0, w_1, \ldots, w_{n-1})^T = (d, 0, \ldots, 0)^T$. Inoltre, dalle ipotesi, abbiamo che $\mathbf{r} = -r\mathbf{e}_1 + \mathbf{e}_2 \in \mathcal{L}(\mathbf{B_v})$ per qualche intero r e possiamo assumere, senza perdita di generalità, che $r \in [-d/2, d/2)$, dato che altrimenti basta sottrarre da $\mathbf{r}$ multipli di $d \cdot \mathbf{e}_1$ fino a quando la condizione risulta soddisfatta:

$$\begin{array}{rcccccc} & (-r & 1 & 0 & \ldots & 0)^T \\ -k\cdot & (\ d & 0 & 0 & \ldots & 0)^T \\ \hline = & ([-r]_d & 1 & 0 & \ldots & 0)^T \end{array}$$

   per un opportuno $k \in \mathbb{Z}$. Per $i = 1, 2, \ldots, n-1$ denotiamo con $r_i \stackrel{\text{def}}{=} [r^i]_d$ e proviamo per induzione che il reticolo $\mathcal{L}(\mathbf{B_v})$ contiene il vettore $\mathbf{r}_i \stackrel{\text{def}}{=} -r_i \cdot \mathbf{e}_1 + \mathbf{e}_{i+1}$. In tal modo, ponendo tutti questi vettori trasposti a formare le righe di una matrice, otteniamo proprio la matrice nella forma cercata. Per $i = 1$, $\mathbf{r}_1 = \mathbf{r}$, che appartiene a $\mathcal{L}(\mathbf{B_v})$ come visto prima. Dimostriamo ora che, se i vettori $\mathbf{r}_i$ appartengono a $\mathcal{L}(\mathbf{B_v})$ per $i \in [1, n-2]$, allora ciò è vero anche per $i+1$. Ricordiamo che il reticolo è chiuso rispetto allo spostamento

circolare degli elementi del vettore; pertanto, se $\mathbf{r}_i = -r_i \cdot \mathbf{e}_1 + \mathbf{e}_{i+1} \in \mathcal{L}(\mathbf{B_v})$, $\mathbf{s}_{i+1} \stackrel{\text{def}}{=} -r_i \cdot \mathbf{e}_2 + \mathbf{e}_{i+2} \in \mathcal{L}(\mathbf{B_v})$ e, di conseguenza, il reticolo contiene anche il vettore

$$\mathbf{s}_{i+1} + r_{i+1} \cdot \mathbf{r} = (-r_i \cdot \mathbf{e}_2 + \mathbf{e}_{i+2}) + r_i(-r\mathbf{e}_1 + \mathbf{e}_2) = -r_i r \cdot \mathbf{e}_1 + \mathbf{e}_{i+2}.$$

Basta ridurre $-r_i r$ modulo d, aggiungendo o sottraendo un multiplo adeguato di $d \cdot \mathbf{e}_1$ e otteniamo il vettore

$$[-r \cdot r_i]_d \cdot \mathbf{e}_1 + \mathbf{e}_{i+2} = -[r^{i+1}]_d \cdot \mathbf{e}_1 + \mathbf{e}_{i+2} = \mathbf{r}_{i+1} \in \mathcal{L}(\mathbf{B_v}),$$

concludendo così la dimostrazione. □

Osserviamo che la forma normale hermitiana viene ad essere individuata da due interi: d e r; quindi la chiave pubblica può essere fornita attraverso i suddetti interi. Si può adottare un argomento analogo per la chiave segreta, che vedremo di seguito nella procedura di decodifica.

4.1.2 Codifica

Per cifrare un bit $b \in \{0, 1\}$ con la chiave pubblica rappresentata dagli interi d, r della matrice in forma hermitiana normale, prima di tutto scegliamo un vettore casuale $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})^T$, in cui ogni elemento $u_i \in \{0, \pm 1\}$ è pari a 0 con una certa probabilità q e ± 1 con probabilità $(1-q)/2$ ciascuno. Definiamo poi il vettore $\mathbf{a} = 2\mathbf{u} + b \cdot \mathbf{e}_1 = (2u_0 + b, 2u_1, \ldots, 2u_{n-1})^T$. Il testo cifrato diventa

$$\mathbf{c} = \mathbf{a} \bmod \mathbf{H} = \mathbf{a} - (\mathbf{H} \times \lfloor \mathbf{H}^{-1} \times \mathbf{a} \rceil) = \mathbf{H} \times \underbrace{(\mathbf{H}^{-1} \times \mathbf{a} - \lfloor \mathbf{H}^{-1} \times \mathbf{a} \rceil)}_{\text{parte frazionaria}} = \mathbf{H} \times \{\mathbf{H}^{-1} \times \mathbf{a}\},$$

dove indichiamo con $\{a\} = a - \lfloor a \rceil$ la parte frazionaria di a. Possiamo tuttavia rappresentare $\mathbf{c}$ solo tramite un intero, pertanto dobbiamo mostrare che quanto affermato è corretto. Ricordiamo innanzitutto che $\mathbf{H}$, e quindi anche $\mathbf{H}^{-1}$, sono in una forma speciale

$$\mathbf{H} = \begin{bmatrix} d & -r & -[r^2]_d & -[r^3]_d & \ldots & -[r^{n-1}]_d \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}, \mathbf{H}^{-1} = \frac{1}{d} \cdot \begin{bmatrix} 1 & r & [r^2]_d & [r^3]_d & \ldots & [r^{n-1}]_d \\ 0 & d & 0 & 0 & \ldots & 0 \\ 0 & 0 & d & 0 & \ldots & 0 \\ 0 & 0 & 0 & d & \ldots & 0 \\ \vdots & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & \ldots & d \end{bmatrix}.$$

Indichiamo $\mathbf{a}$ come $(a_0, a_1, \ldots, a_{n-1})^T$ e sia $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Abbiamo allora che $\mathbf{H}^{-1} \times \mathbf{a} = (s/d, a_1, \ldots, a_{n-1})^T$, con $s = a(r) \bmod d$. Quindi la parte frazionaria di $\mathbf{H}^{-1} \times \mathbf{a}$ è $([a(r)]_d/d, 0, \ldots, 0)^T$ e il testo cifrato diventa

$$\mathbf{c} = \mathbf{H} \times ([a(r)]_d/d, 0, \ldots, 0)^T = ([a(r)]_d, 0, \ldots, 0)^T,$$

che può essere così identificato con un intero $c = [a(r)]_d = \left[b + 2\sum_{i=1}^{n-1} u_i r^i\right]_d$. Abbiamo così messo in evidenza che, per codificare un bit, abbiamo bisogno solo di valutare il polinomio $u(x)$ in r (che perciò rappresenta il "rumore"), moltiplicare il risultato per 2 e aggiungere il bit b, tutto questo modulo d.

Possiamo notare che l'operazione che comporta il costo maggiore dal punto di vista computazionale è la valutazione del polinomio $u(x)$ in r. Essa, svolta mediante la regola di Horner, necessita lo svolgimento di $n-1$ moltiplicazioni (anziché $n(n-1)/2$), mentre se è svolta mediante l'algoritmo di Paterson-Stockmeyer ne richiede $O(\sqrt{n})$ [25], che diventa $O(\sqrt{kn})$ per valutare k polinomi [15].

4.1.3 Decodifica

L'algoritmo di decodifica prende in input l'intero c, che rappresenta il vettore $\mathbf{c} = (c, 0, \ldots, 0)^T$, nonché le matrici $\mathbf{B_v}$ e $\mathbf{B_w}$. Il vettore $\mathbf{a} = 2\mathbf{u} + b \cdot \mathbf{e}_1$ può essere facilmente trovato nel seguente modo

$$\mathbf{a} = \mathbf{c} \bmod \mathbf{B_v} = \mathbf{c} - (\mathbf{B_v} \times \lfloor \underbrace{\mathbf{B_v}^{-1}}_{=\mathbf{B_w}/d} \times \mathbf{c} \rceil) = \mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times \mathbf{c} \right\}, \tag{4.3}$$

e poi viene fornito in output il bit meno significativo del primo elemento di $\mathbf{a}$: $b = a_0 \bmod 2$.

La ragione per cui questa procedura di decodifica funziona correttamente risiede nel fatto che le colonne di $\mathbf{B_v}$, e quindi di $\mathbf{B_w}$, sono vicine all'essere ortogonali l'una con l'altra e quindi la moltiplicazione di un vettore per queste matrici non cambia in modo significativo la sua norma. In particolare osserviamo che $\mathbf{a}$ rappresenta la distanza tra $\mathbf{c}$ e qualche vettore del reticolo $\mathcal{L}(\mathbf{B_v})$, ovvero $\mathbf{c} = \mathbf{B_v} \times \mathbf{y} + \mathbf{a}$, per un certo vettore $\mathbf{y} \in \mathbb{Z}^n$. Allora riprendiamo (4.3)

$$\begin{aligned} \mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times \mathbf{c} \right\} &= \mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times (\mathbf{B_v} \times \mathbf{y} + \mathbf{a}) \right\} = \mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times \mathbf{B_v} \times \mathbf{y} + \frac{\mathbf{B_w}}{d} \times \mathbf{a} \right\} \\ &= \mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times \mathbf{a} \right\}, \end{aligned}$$

dove l'ultima uguaglianza discende dal fatto che $\mathbf{y}$ è un vettore intero. Quindi

$$\mathbf{B_v} \times \left\{ \frac{\mathbf{B_w}}{d} \times \mathbf{a} \right\} = \mathbf{a} = \mathbf{B_v} \times \frac{\mathbf{B_w}}{d} \times \mathbf{a},$$

da cui $\left\{\frac{\mathbf{B_w}}{d} \times \mathbf{a}\right\} = \frac{\mathbf{B_w}}{d} \times \mathbf{a}$ se e solo se ogni elemento di $\frac{\mathbf{B_w}}{d} \times \mathbf{a}$ è minore di $1/2$ in valore assoluto.

Volendo ottimizzare la procedura di decodifica, ricordiamo che $\mathbf{c} = (c, 0, \ldots, 0)^T$ e per quanto visto sopra $[\mathbf{B_w} \times \mathbf{c}]_d = [\mathbf{B_w} \times \mathbf{a}]_d = \mathbf{B_w} \times \mathbf{a}$, dove con $[\cdot]_d$ riferito ad un vettore intendiamo che ogni suo elemento è ridotto modulo d e il risultato è contenuto in $[-d/2, d/2)$. Abbiamo che

$$\begin{aligned}[\mathbf{B_w} \times \mathbf{c}]_d &= [c \cdot (w_0, w_1, \ldots, w_{n-1})^T]_d = ([cw_0]_d, \ldots, [cw_{n-1}]_d)^T \\ &= \mathbf{B_w} \times \mathbf{a} = \mathbf{B_w} \times 2\mathbf{u} + \mathbf{B_w} \times b\mathbf{e}_1 = \mathbf{B_w} \times 2\mathbf{u} + b(w_0, \ldots, w_{n-1})^T,\end{aligned}$$

da cui discende la seguente relazione

$$([cw_0]_d, \ldots, [cw_{n-1}]_d)^T = b(w_0, \ldots, w_{n-1})^T \pmod 2;$$

in altre parole, per ogni $i \in [0, n-1]$, vale $[cw_i]_d = bw_i \bmod 2$ e, pertanto, è sufficiente tenere in considerazione solo uno degli elementi w_i (che deve essere dispari) e con esso calcolare $b = [cw_i]_d \bmod 2$.

4.2 Parte II: Crittosistema pienamente omomorfo

4.2.1 Compressione della decodifica

Quando consideriamo la funzione di decodifica come polinomio nei bit della chiave segreta, il grado è troppo alto per essere utilizzato nei processi valutazione omomorfa dal crittosistema descritto fino ad ora, che pertanto non risulta bootstrappable. Per raggiungere questa proprietà dobbiamo cambiare il formato della chiave segreta e inserire qualche informazione su di essa nella chiave pubblica per ottenere un'espressione della decodifica di grado minore. Pertanto aggiungiamo alla chiave pubblica un insieme di elementi $\{x_i \in \mathbb{Z}_d : i = 1, 2, \ldots, G\}$ tale per cui esista un sottoinsieme molto rado di x_i che sommati diano w modulo d. La chiave segreta acquisterà allora il ruolo di vettore che caratterizza tale sottoinsieme, cioè sarà una sequenza di bit $(\sigma_1, \ldots, \sigma_G)$ avente peso di Hamming pari a $g \ll G$ e $\sum_{i=1}^{G} \sigma_i x_i = \sum_{i=1}^{g} \sigma_i x_i = w \bmod d$. Allora, dato un testo cifrato $c \in \mathbb{Z}_d$, calcoliamo $y_i = \langle cx_i \rangle_d$, dove con $\langle \cdot \rangle_d$ intendiamo la riduzione modulo d con risultato in $[0, d)$, e la funzione di decodifica potrà essere così scritta: $\left[\sum_{i=1}^{G} \sigma_i y_i\right]_d \bmod 2$. Mostriamo che la decodifica può essere espressa da un polinomio di grado minore nei bit σ_i. Innanzitutto abbiamo

$$\left[\sum_{i=1}^{G} \sigma_i y_i\right]_d = \left(\sum_{i=1}^{G} \sigma_i y_i\right) - d \cdot \left\lfloor \frac{\sum_i \sigma_i y_i}{d} \right\rceil = \left(\sum_{i=1}^{G} \sigma_i y_i\right) - d \cdot \left\lfloor \sum_i \sigma_i \frac{y_i}{d} \right\rceil,$$

dopodiché operiamo la riduzione modulo 2 separatamente su ogni termine e applicando poi la disgiunzione esclusiva sui bit risultanti (XOR):

$$\left(\bigoplus_{i=1}^{G}\sigma_i\langle y_i\rangle_2\right)\oplus\langle d\rangle_2\cdot\left\langle\left\lfloor\sum_i\sigma_i\frac{y_i}{d}\right\rceil\right\rangle_2=\bigoplus_{i=1}^{G}\sigma_i\langle y_i\rangle_2\oplus\left\langle\left\lfloor\sum_i\sigma_i\frac{y_i}{d}\right\rceil\right\rangle_2,$$

dove l'ultima uguaglianza segue dal fatto che d è dispari e pertanto $\langle d\rangle_2=1$. Notiamo che le y_i e d sono costanti e che la decodifica è funzione solo dei bit σ_i, quindi $\bigoplus_{i=1}^{G}\sigma_i\langle y_i\rangle_2$ è lineare nei σ_i e l'unico termine non lineare è $\left\langle\left\lfloor\sum_i\sigma_i\frac{y_i}{d}\right\rceil\right\rangle_2$.

Osserviamo tuttavia che è necessario fare attenzione che il rumore non aumenti troppo (abbiamo visto prima che wc/d deve mantenersi minore di $1/2$) e nel caso di un crittosistema bootstrappable la distanza di wc dal multiplo più vicino di d deve risultare inferiore a $d/2(g+1)$, dove g è il numero di bit non nulli nella chiave segreta. Abbiamo

$$\mathsf{abs}([wc]_d)=\mathsf{abs}\left(\left[\sum_{i=1}^{G}\sigma_i y_i\right]_d\right)<\frac{d}{2(g+1)}\quad\Rightarrow\quad\mathsf{abs}\left(\left[\sum_{i=1}^{G}\sigma_i\frac{y_i}{d}\right]_d\right)<\frac{1}{2(g+1)}.\tag{4.4}$$

Dato che $y_i\in[0,d-1]$ per ogni i , risulta $y_i/d\in[0,1)$ e, se consideriamo le approssimazioni z_i di y_i/d a $p=\lceil\log_2(g+1)\rceil$ bit dopo la virgola binaria, abbiamo che $z_i\in[0,1]$ è il numero più vicino a y_i/d rispetto a tutti i numeri della forma $a/2^p$, con a un intero in $[0,2^p]$. Allora

$$\mathsf{abs}\left(z_i-\frac{y_i}{d}\right)\leq 2^{-(p-1)}\leq\frac{1}{2(g+1)}$$

e consideriamo l'effetto della sostituzione degli elementi $\sigma_i y_i/d$ con $\sigma_i z_i$: se $\sigma_i=0$ allora la somma rimane invariata; se $\sigma_i=1$ allora la somma cambia di al massimo $2^{-(p+1)}\leq 1/2(g+1)$. Per quel valore di i, di $\sigma_i=1$ ce ne sono g perciò la somma $\sum_i\sigma_i z_i$ si discosta dalla somma originaria $\sum_i\sigma_i y_i/d$ di al massimo $g/2(g+1)$; unendo questo a quanto visto in (4.4), otteniamo che la somma $\sum_i\sigma_i z_i$ dista dal più vicino intero al massimo $1/2(g+1)+g/2(g+1)=1/2$ e pertanto

$$\left\lfloor\sum_{i=1}^{G}\sigma_i\frac{y_i}{d}\right\rceil=\left\lfloor\sum_{i=1}^{G}\sigma_i z_i\right\rceil.$$

Possiamo concludere che per un testo cifrato c abbastanza vicino al reticolo la funzione di decodifica può essere calcolata nel seguente modo:

$$D_{c,d}((\sigma_1,\ldots,\sigma_G))=\left\langle\left\lfloor\sum_i\sigma_i z_i\right\rceil\right\rangle_2\oplus\bigoplus_i\sigma_i\langle y_i\rangle_2,$$

in cui l'unica parte non lineare è l'addizione e l'arrotondamento (modulo 2) degli z_i, aventi però solo p bit di precisione a destra della virgola binaria.

La somma degli elementi $\sigma_i z_i$ comporta ovviamente un costo computazionale, riportiamo allora l'ottimizzazione di questa procedura che troviamo in [15], chiamato l'algoritmo di addizione della scuola elementare, che richiede g^2 moltiplicazioni, dove g è il numero di $\sigma_i = 1$. Tuttavia esistono altri metodi per implementare questa somma, ad esempio il già citato 3-for2 trick che richiede solo $g \cdot p$ moltiplicazioni. Ciò nonostante Gentry e Halevi hanno scelto comunque il metodo della *grade-school addition* per avere un polinomio di decodifica di grado minore (15 anziché 16) e per ridurre il tempo d'esecuzione richiesto dall'algoritmo (solo il 10% del tempo totale).
Per fare ciò vediamo l'insieme $(\sigma_1, \ldots, \sigma_G)$ come un vettore-riga $\vec{\sigma}$ dal peso di Hamming pari a g e suddividiamolo in g vettori $\vec{\sigma}_1, \ldots, \vec{\sigma}_g$, ognuno contenente un unico 1 in modo tale che sommati diano il vettore originario $\vec{\sigma}$. Inoltre usiamo g diversi insiemi grandi, $B_1, \ldots, B_g$, e ogni vettore $\vec{\sigma}_k$ sceglie un elemento dall'insieme B_k corrispondente, facendo sì che questi g elementi sommati diano w modulo d. Indichiamo i bit di $\vec{\sigma}_k$ con $\sigma_{k,i}$ e gli elementi di B_k con $\{x(k,i) \,:\, i = 1, 2, \ldots, G\}$, inoltre definiamo $y(k,i) = \langle cx(k,i)\rangle_d$ e $z(k,i)$ l'approssimazione di $y(k,i)/d$ con p bit di precisione dopo la virgola binaria. Con queste notazioni possiamo riscrivere la funzione di decodifica come

$$D_{c,d}((\vec{\sigma}_1, \ldots, \vec{\sigma}_G)) = \left\langle \left\lfloor \sum_{k=1}^{g} \left(\sum_{i=1}^{G} \sigma_{k,i} z(k,i) \right) \right\rceil \right\rangle_2 \oplus \bigoplus_{k,i} \sigma_{k,i} \langle y(k,i)\rangle_2. \tag{4.5}$$

Denotando con $q_k = \sum_i \sigma_{k,i} z(k,i)$, per $k = 1, \ldots, g$, osserviamo che ognuno di questi q_k è ottenuto sommando G numeri, al massimo uno dei quali è diverso da zero.

L'algoritmo di addizione della scuola elementare (Grade-school addition algorithm). Una volta ottenuti g numeri con $p = \lceil \log(g+1) \rceil$ bit di precisione nella rappresentazione binaria, possiamo usare il semplice algoritmo della scuola elementare per sommarli, che descriviamo qui di seguito rispetto al nostro contesto. Mettiamo questi numeri in g righe; si vengono a formare $p+1$ colonne (una colonna per ogni bit di posizione, l'indice di colonna sarà da sinistra a destra $0, -1, \ldots, -p$). Ogni colonna è formata da bit, che sommiamo a partire dalla colonna $-p$, e segnamo il riporto sopra la colonna $-p+1$ alla sua sinistra e ricominciamo. In generale, ogni volta che aggiungiamo il riporto, il numero di bit in quella colonna aumenta di uno, perciò verranno sommati rispettivamente prima g bit, poi $g+1, g+2, \ldots, g+p-1$. Quindi il numero di operazioni eseguite durante il processo sono al più $g \cdot 2^{p-1} + \sum_{k=1}^{p-1} (g+k) \cdot 2^{p-k} = O(g^2)$.

4.2.2 Riduzione della dimensione della chiave pubblica

Due fattori contribuiscono in modo significativo a rendere elevata la dimensione della chiave pubblica: il bisogno di sicurezza che chiama in causa lo sparse-subset-sum problem (SSSP) e l'aggiunta nella chiave pubblica di codifiche dei bit della chiave segreta. Al fine di ridurre la dimensione della chiave pubblica dobbiamo lavorare su questi due contributi come viene mostrato nel seguito.

Sparse-subset-sum problem

Dall'ottimizzazione precedente ricordiamo che abbiamo g insiemi $B_1, \ldots, B_g$, ognuno con G elementi di $\mathbb{Z}_d$, tali che esista una collezione di tali elementi, uno per ogni B_k, che sommati diano la chiave segreta w modulo d. Rappresentare tutti questi insiemi esplicitamente richiederebbe l'inserimento di $g \cdot G$ elementi di $\mathbb{Z}_d$ nella chiave pubblica, pertanto ci limitiamo a inserirne solo g, $x_1, \ldots, x_g$, in modo che definiscano implicitamente $B_1, \ldots, B_g$. Ossia, dato B_k, esso contiene gli elementi $x(k,i)$ che sono così ridefiniti: $x(k,i) = \langle x_k \cdot R^i \rangle_d$, per $i = 1, \ldots, G$ e R è un parametro fissato per evitare attacchi allo SSSP. Si ha che esiste un solo indice i_k in ogni insieme tale che $\sum_k x(k, i_k) = w \bmod d$.

Codifiche della chiave segreta

La chiave segreta del sistema compresso consiste di g vettori composti da G bit ciascuno, tali che solo uno di questi bit sia uguale a 1 e che gli altri siano tutti nulli. Se codifichiamo ciascuno di questi bit individualmente, dovremmo includere nella chiave pubblica $g \cdot G$ testi cifrati, ognuno appartenente a $\mathbb{Z}_d$. Invece cerchiamo di includere una loro rappresentazione implicita che prenda meno spazio ma che permetta comunque di avere la codifica di tutti questi bit. Dato che il sistema è parzialmente omomorfo, è possibile fornire per ogni B_k una descrizione codificata della funzione che sull'input i risulta 1 se e solo se $i = i_k$. Una tale funzione può essere rappresentata usando $\log G$ bit (il numero di bit che servono a rappresentare i_k) ed espressa come un polinomio di grado $\log G$ in questi bit. Quindi, in linea di principio, è possibile rappresentare la codifica di tutti i bit della chiave segreta usando solo $g \log G$ testi cifrati. Purtroppo però insorgono dei problemi in relazione al grado dei polinomi supportato dal sistema (che in questo modo è cresciuto) e al tempo d'esecuzione, anch'esso aumentato vanificando l'ottimizzazione svolta nella sezione precedente. In particolare dall'equazione (4.5), la decodifica era divenuta $D_{c,d}((\vec{\sigma}_1, \ldots, \vec{\sigma}_G)) = \left\langle \left\lfloor \sum_{k=1}^{g} \left(\sum_{i=1}^{G} \sigma_{k,i} z(k,i) \right) \right\rceil \right\rangle_2 \oplus \bigoplus_{k,i} \sigma_{k,i} \langle y(k,i) \rangle_2$. Dato che la codifica di

ognuno dei bit $\sigma_{k,i}$ ora è un polinomio di grado $\log G$, abbiamo bisogno che il crittosistema sia in grado di supportare polinomi di grado aumentato di $\log G$ volte, rispetto alla situazione in cui avessimo fornito direttamente i $\sigma_{k,i}$; inoltre, a proposito del tempo d'esecuzione, laddove prima il calcolo dei $q_k = \sum_{i=1}^{G} \sigma_{k,i} z(k,i)$ richiedeva solo addizioni, ora necessita di $G \log G$ moltiplicazioni per determinare tutti i $\sigma_{k,i}$.

Per questo motivo scegliamo una strada differente, che permette l'inserimento nella chiave pubblica di un numero inferiore di testi cifrati, $O(\sqrt{G})$ per ogni insieme B_k, e l'esecuzione di $p\sqrt{G}$ moltiplicazioni per ogni q_k. Nello specifico, per ogni insieme B_k, teniamo nella chiave pubblica un numero $\kappa > \lceil \sqrt{2G} \rceil$ di testi cifrati, di cui tutti tranne due sono codifiche di zero. Allora la codifica di ogni bit $\sigma_{k,i}$ della chiave segreta è ottenuta moltiplicando due di questi testi cifrati. Sia $\binom{\kappa}{2}$ il numero di coppie distinte prese nell'intervallo $[1,\kappa]$ e, per ogni coppia di interi $a \neq b \in [1,\kappa]$, denotiamo con $i(a,b)$ l'indice della coppia (a,b) in ordine lessicografico su $\binom{\kappa}{2}$:

$$i(a,b) \stackrel{\text{def}}{=} (a-1) \cdot c - \binom{a}{2} + (b-a).$$

In particolare, se a_k, b_k sono gli indici di due codifiche di 1 nel gruppo corrispondente all'insieme B_k, allora $i_k = i(a_k, b_k)$. Usiamo un'implementazione che calcoli prima la somma e poi la moltiplicazione di due testi cifrati, anziché il contrario; in termini concreti siano $\{\eta_m^{(k)} : k \in [1,g], m \in [1,\kappa]\}$ i bit la cui codifica è contenuta nella chiave pubblica (per ogni k esattamente due $\eta_m^{(k)}$ sono uguali a 1, mentre gli altri sono tutti nulli, per cui ogni $\sigma_{k,i}$ è ottenuto come prodotto di questi due $\eta_m^{(k)}$):

$$q_k = \sum_{a,b} \underbrace{\eta_a^{(k)} \eta_b^{(k)}}_{\sigma(k,i(a,b))} z(k, i(a,b)) = \sum_a \eta_a^{(k)} \sum_b \eta_b^{(k)} z(k, i(a,b)). \tag{4.6}$$

Dato che abbiamo i bit in chiaro di $z(k,i(a,b))$, possiamo codificare i bit di $\eta_b^{(k)} z(k,i(a,b))$ moltiplicando la codifica di $\eta_b^{(k)}$ per 0 o per 1. Le uniche vere moltiplicazioni in $\mathbb{Z}_d$ che dobbiamo eseguire sono quelle per $\eta_a^{(k)}$ ed esse sono $O(p\sqrt{G})$ per ogni q_k. Notiamo che al variare del parametro κ, cioè del numero di testi cifrati contenuti nella chiave pubblica per ogni insieme B_k, cambia la strada stessa intrapresa; κ deve essere maggiore di $\lceil \sqrt{2G} \rceil$ per essere in grado di codificare ogni indice $i \in [1,G]$ tramite una coppia (a,b) tra le $\binom{\kappa}{2}$ possibili, ma, se prendiamo valori di κ maggiori, diminuisce il numero di moltiplicazioni nel calcolo espresso dall'Equazione (4.6), a costo dell'aumento della lunghezza della chiave pubblica. Gentry in [15] ha posto $\kappa = \lceil 2\sqrt{G} \rceil$.

4.3 I parametri

- **I parametri di sicurezza λ e μ.**
  Ci sono due parametri di sicurezza che guidano la scelta di tutti gli altri: λ (che controlla la complessità degli attacchi al crittosistema, quali la ricerca esaustiva o l'attacco del compleanno) e μ (relativo al Bounded Distance Decoding Problem, BDDP). Possiamo considerare $\lambda = 72$, che significa che ci possono essere attacchi del compleanno con complessità 2^{72}. Per quanto riguarda il parametro μ, esso quantifica la difficoltà del problema del vettore più corto (SVP) e del BDDP. Nello specifico prendiamo tre possibili valori di $\mu \approx 2.17, \mu \approx 0.54$ o $\mu \approx 0.14$ corrispondenti a reticoli di dimensione $n = 2^{11}, n = 2^{13}$ o $n = 2^{15}$.

- **La dimensione n.**
  Come accennato sopra scegliamo la dimensione del reticolo sufficientemente grande per aumentare la complessità e quindi la sicurezza del crittosistema: $n \in \{2^{11}, 2^{13}, 2^{15}\}$.

- **Le dimensioni $g \ll G$ e il parametro di precisione p.**
  Un altro parametro che influenza notevolmente l'implementazione è la dimensione g del sottoinsieme di elementi scelti da un insieme molto più grande, di dimensione G, aggiunto alla chiave pubblica. Relativamente allo Sparse Subset Sum Problem (SSSP), consideriamo $g = 15$ e, dato che $p = \lceil \log(g + 1) \rceil$, il numero di bit di precisione scelto è $p = 4$, che significa che rappresentiamo ogni $z(k, i)$ con 4 bit di precisione a destra della virgola binaria e un bit a sinistra. Per quanto riguarda il valore di G, invece, sappiamo che gli attacchi col metodo di forza bruta allo SSSP richiedono un tempo d'esecuzione di $G^{\lceil g/2 \rceil}$ e e perciò deve valere $G^{\lceil g/2 \rceil} \geq 2^{\lambda}$, che in base a come abbiamo fissato gli altri parametri ($g = 15$ e $\lambda = 72$ diventa $G^8 \geq 2^{72}$ e $G \geq 512$.

4.4 I risultati dell'implementazione

Con questi parametri l'implementazione realizzata da Gentry e Halevi ha fornito i risultati riportati nella Figura 4.1; l'elaborazione è stata effettuata con una macchina dalle seguenti specifiche: IBM System x3500 server, processore Intel Xeon E5450 quad-core a 64-bit, 32 GHz, 12MB L2 cache e 24 GB di RAM.

Dimension n	SK bit-size t	# of ctxts in PK ($g\kappa$)	PK size $\approx g\cdot\kappa\cdot\lvert d\rvert$	keyGen	Recrypt
512	380	690	17 MByte	2.5 sec	6 sec
2048	380	690	69 MByte	41 sec	32 sec
8192	380	705	284 MByte	8.4 min	2.8 min
32768	380	1410	2.25 GByte	2.2 hour	31 min

Figura 4.1: Risultati in termini di tempo di esecuzione ottenuti considerando un parametro di sicurezza di 72 bit. Specifichiamo che il termine d indica il grado del polinomio di decodifica compressa, qui è stato preso pari a 15.

Il crittosistema originale di Gentry basato sui reticoli-ideali, descritto nel Capitolo 3, ha avuto seguito e, grazie alle ottimizzazioni svolte da Gentry stesso insieme a Halevi in [15], è stato sviluppato. Tuttavia, come possiamo notare dalla Figura 4.1, abbiamo dei dati che compromettono in modo significativo il suo utilizzo nella realtà: la dimensione della chiave pubblica (2.25 GB) e il tempo necessario alla procedura di Recrypt (31 minuti). Sono valori piuttosto alti, considerando anche le elevate prestazioni della workstation utilizzata; ad esempio Gentry ha stimato che una semplice ricerca su Google fatta con parole codificate richiederebbe un tempo pari a milioni di volte quello attuale.
Facciamo un confronto con il crittosistema descritto nel Capitolo 1; anch'esso è pienamente omomorfo, ma è concettualmente più semplice, perché opera sugli interi piuttosto che sui reticoli-ideali. Recentemente è stata implementata una sua variante ottimizzata ([8]), in cui è stata ridotta la dimensione della chiave pubblica; ne riportiamo i risultati in Figura 4.2. L'implementazione è stata effettuata con un computer Intel Core2 Duo E8500 CPU a 3.12 GHz.

Parameters	KeyGen	Encrypt	Expand	Decrypt	Recrypt	pk size
Toy λ=42	4.38 s	0.05 s	0.03 s	0.01 s	1.92 s	0.95 MB
Small 52	36 s	0.79 s	0.46 s	0.01 s	10.5 s	9.6 MB
Medium 62	5 min 9 s	10 s	8.1 s	0.02 s	1 min 20 s	89 MB
Large 72	43 min	2 min 57 s	3 min 55 s	0.05 s	14 min 33 s	802 MB

Figura 4.2: Risultati in termini di tempo di esecuzione ottenuti per l'implementazione del crittosistema pienamente omomorfo basato sull'aritmetica modulare.

Possiamo osservare che ci sono stati dei miglioramenti, per quanto i tempi siano ancora troppo elevati per una realizzazione pratica; tuttavia, a partire dalla tesi di Gentry, sono stati

fatti grandi progressi negli ultimi anni. Dedichiamo il prossimo capitolo alla presentazione degli ultimi sviluppi in merito.

4.5 Esempio

Consideriamo un caso semplice, potremmo dire ingenuo, che possa però essere d'aiuto nella comprensione del crittosistema.
Sia $R = \mathbb{Z}/(f(x))$, con $f(x) = x^2 + 1$ e seguiamo i passi fatti in questo capitolo.

Generazione delle chiavi:

1. Sia $v(x) = x + 1$, da cui $\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, e calcoliamo la base di rotazione $\mathbf{B_v}$ del reticolo generato da $(\mathbf{v})$; per farlo ci serve $v(x)x \bmod f(x) = (x^2 + x) \bmod f(x) = x - 1$. Si ottiene $\mathbf{B_v} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$.

2. Troviamo $w(x)$ tale che $w(x)v(x) = \det(\mathcal{L}(\mathbf{B_v}))$. Innanzitutto calcoliamo il determinante del reticolo generato da $(\mathbf{v})$ (possiamo farlo tramite il Teorema 2.5), da cui si ha immediatamente che $\det(\mathcal{L}(\mathbf{B_v})) = |\det(\mathbf{B_v})| = 2$. In alternativa lo stesso calcolo si può effettuare mediante la definizione, $\det(\mathcal{L}(\mathbf{B_v})) = \prod_{i=1}^{2} ||\mathbf{b}_i^*||$, dove $\mathbf{b}_i^* = b_i - \sum_{j=1}^{i-1} \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*$ sono i vettori della base ortogonalizzata. Nel nostro caso i vettori della base $\mathbf{B_v}$ sono già ortogonali[1], quindi $\prod_{i=1}^{2} ||\mathbf{b}_i^*|| = \prod_{i=1}^{2} ||\mathbf{b}_i|| = \sqrt{2} \cdot \sqrt{2} = 2$. Allora $w(x) = -x + 1$, così $w(x)v(x) = 2 \bmod f(x)$, e la base di rotazione del reticolo generato da $(\mathbf{w})$ è ottenuta calcolando $w(x)x \bmod f(x) = x + 1$, da cui possiamo scrivere $\mathbf{B_w} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$. Deve risultare $\mathbf{B_w}\mathbf{B_v} = 2\mathbf{I} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$.

3. Calcoliamo, infine, la forma normale hermitiana di $\mathbf{B_v}$, $\mathrm{HNF}(\mathbf{B_v}) = \mathbf{H} = \begin{bmatrix} d & -r \\ 0 & 1 \end{bmatrix}$, dove r è una radice di $f(x) = 0 \bmod d$. Abbiamo pertanto $\mathbf{H} = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix}$.
La chiave pubblica pk è data dalla coppia $(d, r) = (2, 1)$.

Codifica:

[1] Osserviamo che, se $n = 2$ e $f(x) = x^n + 1$, la base di rotazione è sempre composta da vettori ortogonali tra loro: $\left\langle \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}, \begin{bmatrix} -v_1 \\ v_0 \end{bmatrix} \right\rangle = 0$.

1. Sia $\mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ un vettore a coefficienti in $\{0, \pm 1\}$, corrispondente all'aggiunta di "errore", e sia $b \in \{0, 1\}$ il bit da codificare. Calcoliamo $\mathbf{a} = 2\mathbf{u} + b\mathbf{e}_1 = \begin{bmatrix} 2+b \\ 0 \end{bmatrix}$.

2. La codifica di b è la seguente

$$\mathbf{c} = \mathbf{a} \bmod \mathbf{H} = \mathbf{a} - \mathbf{H}\lfloor \mathbf{H}^{-1}\mathbf{a} \rceil = \mathbf{H}(\mathbf{H}^{-1}\mathbf{a} - \lfloor \mathbf{H}^{-1}\mathbf{a} \rceil)$$

dove $\mathbf{H}^{-1} = \frac{1}{d}\begin{bmatrix} 1 & r \\ 0 & d \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 1 \end{bmatrix}$.
Se $b = 1$, si ha $\mathbf{c} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ o, in alternativa, $\mathbf{c} = \begin{bmatrix} c \\ 0 \end{bmatrix}$ con $c = [2u(r)+b]_2 = -1 \in [-1, 1)$.
Se $b = 0$, si ha $\mathbf{c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}$ con $c = [2u(r) + b]_2 = 0 \in [-1, 1)$.

Decodifica:

$$\mathbf{a} = \mathbf{B_v} \times \left(\frac{\mathbf{B_w}}{d} \times \mathbf{c} - \left\lfloor \frac{\mathbf{B_w}}{d} \times \mathbf{c} \right\rceil \right)$$

oppure $b = [c \cdot w_i]_d \bmod 2$. Se $c = -1$ allora $b = -1 \bmod 2 = 1$, se $c = 0$ allora $b = 0 \bmod 2 = 0$.

Compatibilità con le operazioni di somma e moltiplicazione:
Prendiamo due testi cifrati, c_1 e c_2, ottenuti come sopra, e verifichiamo i casi significativi:

$$c_1 + c_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

che corrisponde a $b_1 + b_2 = 1 + 0 = 0$.

$$c_1 + c_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \pmod 2$$

che corrisponde a $b_1 + b_2 = 1 + 1 = 0 \pmod 2$.

$$c_1 \cdot c_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

che corrisponde a $b_1 \cdot b_2 = 1 \cdot 0 = 0$.

$$c_1 \cdot c_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

che corrisponde a $b_1 \cdot b_2 = 1 \cdot 1 = 1$.

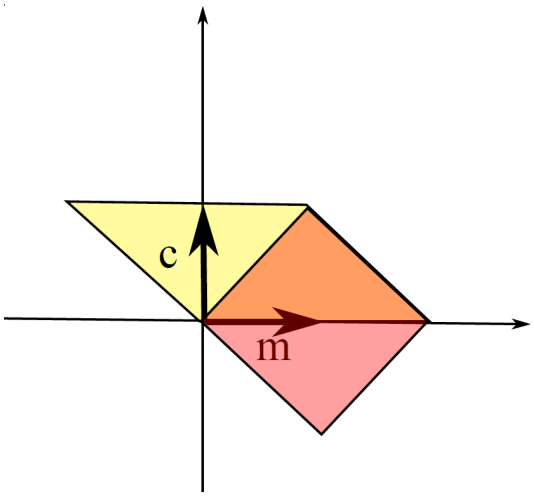

Figura 4.3: In rosso il dominio fondamentale relativo alla base segreta e in giallo alla base pubblica.

Capitolo 5

Ultimi sviluppi

Da quanto abbiamo visto nel precedente capitolo, i tempi di esecuzione sono piuttosto elevati; in particolare il dato maggiormente preoccupante è il tempo necessario all'algoritmo Recrypt per ricodificare un bit (dalla Figura 4.1, 31 minuti). Per questo motivo si sono cercate soluzioni alternative volte a migliorare l'efficienza del crittosistema, tentando nuovi approcci alla fase di transizione da un crittosistema solo parzialmente omomorfo a uno pienamente omomorfo. Come abbiamo visto in §3.7-3.8, Gentry ha utilizzato la tecnica del "bootstrapping", rendendo teoricamente possibile la crittografia omomorfa, tuttavia di recente sono state mostrate altre vie risolutive. Ci stiamo riferendo in particolare alle due varianti seguenti: una che realizza l'omomorfia senza bootstrapping [5] (BGV) e l'altra che la ottiene senza la compressione del circuito di decodifica (fase che in precedenza era necessaria perché il sistema diventasse bootstrappable) [16].

Facciamo il punto di quanto detto finora. Un crittosistema pienamente omomorfo permette di ricevere dati codificati, svolgere operazioni arbitrarie su di essi senza avere la chiave di decodifica e ottenere un risultato coerente, ovvero quando il messaggio codificato viene decifrato, esso corrisponde al risultato delle stesse operazioni svolte sui dati in chiaro. Prima di queste recenti scoperte, tutti i crittosistemi pienamente omomorfi si basavano sulla costruzione originale di Gentry, da noi descritta nel Capitolo 3 e che riassumiamo di seguito. Per prima cosa si costruisce un crittosistema parzialmente omomorfo, ossia capace di compiere operazioni non arbitrariamente lunghe sui testi cifrati, nel senso che sono ammesse solo fino ad un certo numero (le operazioni sono espresse in termini di polinomi di grado limitato o circuiti di profondità limitata). Questo perché i testi cifrati sono affetti da un rumore (ricordiamo che siamo nell'ambito della crittografia probabilistica), che cresce leggermente durante le operazioni di

addizione e molto più significativamente con le moltiplicazioni. A un certo punto il rumore è tale da non permettere più di risalire al testo in chiaro corrispondente, inficiando la correttezza della procedura di decodifica. Per poter realizzare la piena omomorfia è necessario quindi "pulire" i testi cifrati prima di procedere con ulteriori operazioni omomorfe. Gentry ha realizzato questo passaggio tramite la "bootstrappabilità", ossia la proprietà di un crittosistema di essere in grado di valutare una versione leggermente estesa del proprio circuito di decodifica. In pratica il rumore viene ridotto tramite la valutazione del circuito di decodifica sul testo cifrato usando una chiave segreta a sua volta codificata. Pertanto è opportuno comprimere il circuito di decodifica, facendo in modo che rimanga invariata la capacità del circuito, ma sia adeguatamente semplice da permettere la fase di bootstrapping. Gentry ha fatto ciò aggiungendo qualche informazione della chiave segreta all'interno della chiave pubblica, mantenendo il sistema sicuro grazie allo "sparse subset sum problem" (SSSP). Purtroppo tutto questo è realizzabile a discapito dell'efficienza del metodo. Come abbiamo visto dai risultati ottenuti dall'implementazione svolta da Gentry e Halevi la procedura richiede troppo tempo, rendendo il crittosistema inapplicabile alle esigenze reali. Recentemente però Gentry insieme a Halevi e Brakerski con Vaikuntanathan hanno indipendentemente trovato delle soluzioni per costruire un crittosistema pienamente omomorfo senza ricorrere alla fase di compressione e, quindi, senza l'assunzione dello SSSP. I sistemi risultanti costituiscono le maggiori innovazioni rispetto al crittosistema proposto in origine da Gentry.

5.1 Il crittosistema BGV

La grande innovazione fornita da Brakerski e Vaikuntanathan, usata poi in [5], consiste nella creazione di una nuova tecnica per trattare il rumore detta "modulus switching". In BV viene usata una volta sola e poi viene applicata la procedura di bootstrappabilità, mentre in BGV viene applicata iterativamente per mantenere il rumore costantemente a livelli bassi, evitando la fase di bootstrapping. Il metodo adottato in BGV prende il nome di "noise-management technique".
Recentemente questa tecnica è stata applicata anche al crittosistema basato sull'aritmetica modulare del Capitolo 1 da Coron, Naccache e Tibouchi [9].

Procediamo ora con un accenno alla costruzione alla base del crittosistema BGV. Innanzitutto fissiamo un anello R; in concreto esso sarà $\mathbb{Z}$ o $\mathbb{Z}[x]/(x^d + 1)$ con d una potenza di 2.

Specifichiamo che gli elementi dell'anello saranno denotati con lettere minuscole ($r \in R$), mentre i vettori in grassetto ($\mathbf{v} \in R^n$). Indichiamo con $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} u_i \cdot v_i \in R$, l'usuale prodotto scalare tra due vettori $\mathbf{u}, \mathbf{v} \in R^n$, i cui i-esimi coefficienti sono u_i e v_i, rispettivamente. Il crittosistema si basa sul problema detto "Learning with Errors" (LWE), introdotto da Regev nel 2005 [27].

Definizione 5.1. *Fissato un parametro di sicurezza λ, siano n un intero positivo, q un intero ≥ 2 e χ una distribuzione su $\mathbb{Z}$, tutti valori dipendenti da λ. Il problema* $\mathrm{LWE}_{n,q,\chi}$ *consiste nel riuscire a distinguere le seguenti distribuzioni:*

1. *si sceglie il vettore riga $(\mathbf{a}, b)$ in modo casuale in $\mathbb{Z}_q^{n+1}$;*

2. *si sceglie prima il vettore $\mathbf{z}$ in modo casuale in $\mathbb{Z}_q^n$, poi il vettore $\mathbf{a}$ in $\mathbb{Z}_q^n$, sempre in modo casuale. Si prende e dalla distribuzione χ e si pone $b_i = \langle \mathbf{a}, \mathbf{s} \rangle + e$. Il risultato fornito è $(\mathbf{a}, b)$.*

Ricordando che una distribuzione sugli interi si dice B-limitata se $\Pr_{e \leftarrow \chi}[|e| > B]$ è trascurabile, possiamo mettere in risalto il legame tra LWE e il problema del vettore più corto. Per un intero positivo n, siano q un intero primo (che dipende da n) e $B \geq 2n$. Consideriamo una distribuzione χ B-limitata tale che se esiste un algoritmo che risolva efficientemente $\mathrm{LWE}_{n,q,\chi}$ allora esiste un algoritmo per risolvere una versione approssimata del problema del vettore indipendente più corto ($\widetilde{O}(qn^{3/2}/B)$-SIVP).
Diamo un esempio chiarificatore della versione di ricerca del problema LWE. Esso chiede di trovare $\mathbf{s} \in \mathbb{Z}_q^n$ data una sequenza di equazioni lineari approssimate in $\mathbf{s}$:

$$\begin{aligned} 14s_1 + 15s_2 + 5s_3 + 2s_4 &\approx 8 \bmod 17 \\ 13s_1 + 14s_2 + 14s_3 + 6s_4 &\approx 16 \bmod 17 \\ 6s_1 + 10s_2 + 13s_3 + s_4 &\approx 3 \bmod 17 \\ &\vdots \\ 3s_1 + 6s_2 + 4s_3 + 5s_4 &\approx 16 \bmod 17, \end{aligned}$$

dove ogni equazione è corretta a meno di un piccolo errore additivo (diciamo ± 1). Ci sono altre due versioni del LWE: il "Ring Learning with Errors" (RLWE) e il "General Learning with Errors" (GLWE).

Definizione 5.2. *Fissato un parametro di sicurezza λ, siano $f(x) = x^d + 1$, dove d è una potenza di 2, e q un primo ≥ 2. Consideriamo una distribuzione χ su $R_q = \mathbb{Z}_q[x]/(x^d + 1)$. Il problema* $\mathrm{RLWE}_{d,q,\chi}$ *consiste nel saper distinguere le seguenti due distribuzioni:*

1. *si sceglie (a, b) uniformemente da R_q^2;*
2. *si sceglie prima s da R_q in modo uniforme, poi $a \xleftarrow{\mathcal{R}} R_q$. Si prende e dalla distribuzione χ e si pone $b = a \cdot s + e$. Il risultato fornito è $(a, b) \in R_q^2$.*

Il problema RLWE risulta molto utile perché il problema del vettore più corto (SVP) può essere ricondotto ad esso.
I problemi LWE e RLWE sono sintatticamente identici, tranne che nel primo abbiamo usato $\mathbb{Z}$ e nel secondo un anello dei polinomi. Per unificare la trattazione introduciamo il problema General Learning with Errors (GLWE).

Definizione 5.3. *Fissato un parametro di sicurezza λ, siano n un intero positivo, $f(x) = x^d + 1$, dove d è una potenza di 2, e q un primo ≥ 2. Consideriamo una distribuzione χ su $R_q = \mathbb{Z}_q[x]/(x^d + 1)$. Il problema* $\mathrm{GLWE}_{n,f,q,\chi}$ *consiste nel saper distinguere le seguenti due distribuzioni:*

1. *si sceglie (a, b) uniformemente da R_q^{n+1};*
2. *si sceglie prima s da R_q^n in modo uniforme, poi $a \xleftarrow{\mathcal{R}} R_q^n$. Si prende e dalla distribuzione χ e si pone $b = \langle a \cdot s \rangle + e$. Il risultato fornito è $(a, b) \in R_q^{n+1}$.*

LWE è il GLWE con $d = 1$ e RLWE è il GLWE con $n = 1$. Possiamo adesso descrivere il crittosistema basato sul GLWE:

- Configurazione(λ, μ): Usiamo un bit $b \in \{0, 1\}$ per determinare se stiamo facendo riferimento al problema LWE o RLWE, in modo che d e n dipendano non solo dal parametro di sicurezza ma anche dal bit b. Il primo q è lungo μ bit. $d = d(\lambda, \mu, b)$, $n = n(\lambda, \mu, b)$, $N = \lceil (2n + 1) \log q \rceil$, $\chi = \chi(\lambda, \mu, b)$. Sia $R = \mathbb{Z}[x]/(x^d + 1)$ e $\mathrm{param} = (q, d, n, N, \chi)$.
- Chiave segreta(param): Come coefficienti di $\mathbf{s}'$ prendiamo n elementi scelti da χ; poniamo $\mathsf{sk} = \mathbf{s} = (1, \mathbf{s}')^T \in R_q^{n+1}$.
- Chiave pubblica(param, sk): Generiamo la matrice $\mathbf{A}' \xleftarrow{R} M_{N \times n}(R_q)$, un vettore $\mathbf{e}$ formato da N coefficienti presi da χ e poniamo $\mathbf{b} \leftarrow \mathbf{A}'\mathbf{s}' + 2\mathbf{e}$. Sia $\mathbf{A}$ una matrice con $n + 1$

colonne che consistono del vettore $\mathbf{b}$ seguite dalle n colonne di $-\mathbf{A}'$, risulta $\mathbf{As} = 2\mathbf{e}$. La chiave pubblica è $\mathsf{pk} = \mathbf{A}$.

- Codifica(param,pk, m): Codifichiamo un messaggio $m \in \{0,1\}$, ponendo come prima cosa $\mathbf{m} = (m, 0, \ldots, 0) \in R_q^{n+1}$, poi si sceglie un vettore $\mathbf{r} \in \{0,1\}^N$. Il risultato è $\mathbf{c} = \mathbf{m} + \mathbf{A}^T\mathbf{r} \in R_q^{n+1}$.

- Decodifica(param,$\mathsf{sk}, \mathbf{c}$): $m = (\langle \mathbf{c}, \mathbf{s} \rangle \bmod q) \bmod 2$, in cui la riduzione modulo q fornisce un risultato nell'intervallo $(-q/2, q/2]$.

Modulus-Switching. L'essenza della tecnica modulus-switching è trasformare un testo cifrato modulo q in un altro testo cifrato che modulo p mantenga inalterata la correttezza: $\langle \mathbf{c}', \mathbf{s} \rangle \bmod p = \langle \mathbf{c}, \mathbf{s} \rangle \bmod q$. Se p è adeguatamente minore di q, il rumore decresce, ovvero $|\langle \mathbf{c}', \mathbf{s} \rangle \bmod p| = |\langle \mathbf{c}, \mathbf{s} \rangle \bmod q|$. Ricordiamo che $\mathbf{c}, \mathbf{s} \in R_q^{n+1}$ e la riduzione mod q fornisce il risultato in $(-q/2, q/2]$, come mod p in $(-p/2, p/2]$.

Lemma 5.4. *Siano p e q due interi positivi dispari e sia $\mathbf{c}$ è un vettore a coefficienti interi. Definiamo $\mathbf{c}'$ come il vettore più vicino a $(p/q)\mathbf{c}$ tale che $\mathbf{c}' = \mathbf{c} \bmod 2$. Allora, per ogni $\mathbf{s}$ con $|\langle \mathbf{c}, \mathbf{s} \rangle \bmod q| < q/2 - (q/p)l_1(\mathbf{s})$, abbiamo*

$$\begin{aligned} \langle \mathbf{c}', \mathbf{s} \rangle \bmod p &= \langle \mathbf{c}, \mathbf{s} \rangle \bmod q \bmod 2, \\ |\langle \mathbf{c}', \mathbf{s} \rangle \bmod p| &< (p/q)|\langle \mathbf{c}, \mathbf{s} \rangle \bmod q| + l_1(\mathbf{s}), \end{aligned}$$

dove $l_1(\mathbf{s})$ è la norma l_1 di $\mathbf{s}$.

Dimostrazione. Per qualche intero k, abbiamo $\langle \mathbf{c}, \mathbf{s} \rangle \bmod q = \langle \mathbf{c}, \mathbf{s} \rangle - kq$. Per lo stesso k, sia $e_p = \langle \mathbf{c}', \mathbf{s} \rangle - kp \in \mathbb{Z}$. Dato che $\mathbf{c}' = \mathbf{c} \bmod 2$ e $p = q \bmod 2$, abbiamo $e_p = (\langle \mathbf{c}, \mathbf{s} \rangle \bmod q) \bmod 2$. Quindi, per provare il lemma, basta mostrare che $e_p = \langle \mathbf{c}', \mathbf{s} \rangle \bmod p$ e che ha norma adeguatamente piccola. Abbiamo $e_p = (p/q)(\langle \mathbf{c}, \mathbf{s} \rangle \bmod q) + \langle \mathbf{c}' - (p/q)\mathbf{c}, \mathbf{s} \rangle$, e quindi $|e_p| \leq (p/q)(\langle \mathbf{c}, \mathbf{s} \rangle \bmod q) + l_1(\mathbf{s}) < p/2$. L'ultima disuguaglianza implica che $e_p = \langle \mathbf{c}', \mathbf{s} \rangle \bmod p$. □

Esempio. Diamo un esempio che illustri come il rumore diminuisca dopo aver effettuato la tecnica del Modulus-Switching. Siano $q = 127$, $p = 29$, $\mathbf{c} = (175, 212)$ e $\mathbf{s} = (2, 3)$. Abbiamo $\langle \mathbf{c}, \mathbf{s} \rangle \bmod q = 986 - 8 \cdot 127 = -30$. Troviamo $\mathbf{c}'$, il vettore più vicino a $(p/q)\mathbf{c}$ tale che

$\mathbf{c}' = \mathbf{c} \bmod 2$:

$$\begin{aligned} \frac{p}{q} \cdot \mathbf{c} &\approx (39.9, 48.4) \\ \mathbf{c}' &= \mathbf{c} \bmod 2 = (39, 48). \end{aligned}$$

Allora

$$\langle \mathbf{c}', \mathbf{s} \rangle \bmod p = 222 - 8 \cdot 29 = -10 = -30 \bmod 2$$

e il valore assoluto del rumore è diminuito da 30 a 10.

5.2 Il crittosistema chimerico

L'innovazione fornita da Gentry e Halevi in [16] consiste in un nuovo modo di valutare il circuito di decodifica di un crittosistema per diminuire il rumore dei testi cifrati e passare da un crittosistema parzialmente omomorfo a uno pienamente omomorfo. La decodifica viene espressa da un polinomio e la valutazione dei polinomi è fatta tramite i circuiti booleani. Invece è possibile sfruttare l'osservazione di Ben-Or, secondo la quale i polinomi simmetrici multilineari possono essere calcolati da circuiti aritmetici ($\Sigma\Pi\Sigma$) di profondità 3 su $\mathbb{Z}_p$, dove p è un primo sufficientemente grande. Si parla in questo caso di un crittosistema chimerico (Chimeric FHE). L'obiettivo è costruire un sistema che sia bootstrappable, ma il problema è che le moltiplicazioni aumentano molto il rumore. La soluzione proposta consiste nel costruire un crittosistema ibrido, combinando il crittosistema ElGamal, che è efficace per la parte moltiplicativa, con il crittosistema parzialmente omomorfo basato sui reticoli (SWHE), che invece supporta bene le addizioni. Esprimiamo la funzione di decodifica come un circuito aritmetico di profondità 3, in cui sono state raccolte le operazioni in modo che venga svolta prima l'addizione, poi tutte le moltiplicazioni a un unico livello e infine ancora le addizioni, contrassegnate proprio con i simboli delle rispettive operazioni ($\Sigma\Pi\Sigma$). Le somme sono svolte dal crittosistema parzialmente omomorfo, mentre le moltiplicazioni sono effettuate da un crittosistema moltiplicativo, quale ElGamal. Per passare da un crittosistema ad un altro, basta valutare la funzione di decodifica di ElGamal in modo omomorfo sui testi cifrati da esso ottenuti, ottenendo così dei testi cifrati valutabili dal crittosistema originario SWHE.

Per costruire un crittosistema chimerico il crittosistema parzialmente omomorfo utilizzato deve essere compatibile con quello moltiplicativo scelto. Prima di precisare che cosa intendiamo con l'appellativo compatibile, diamo la definizione di circuito ristretto.

Definizione 5.5. *Sia $\mathcal{P}$ l'insieme dei polinomi $P_j(x_1, \ldots, x_n)$ in n variabili. Un circuito aritmetico C viene detto* ***circuito $\mathcal{P}$-ristretto*** *di profondità 3 se esistono dei sottoinsiemi $S_1, \ldots, S_t \subseteq \mathcal{P}$ e delle costanti $\lambda_0, \lambda_1, \ldots, \lambda_t$ tali che il polinomio calcolato dal circuito C sia*

$$\lambda_0 + \sum_{i=1}^{t} \lambda_i \cdot \prod_{P_j \in S_i} P_j(x_1, \ldots, x_n).$$

Chiamiamo $d = \max_i |S_i|$ il $\mathcal{P}$-grado di C.

Possiamo ora definire che cosa significa compatibile.

Definizione 5.6. *Indichiamo il crittosistema parzialmente omomorfo con SWHE e quello moltiplicativo con MHE. Lo spazio dei messaggi in chiaro di SWHE sia $\mathbb{Z}_p$ e la classe di polinomi valutati correttamente da SWHE sia $\mathcal{F}$. Mentre per MHE lo spazio dei testi in chiaro sia $\mathcal{M} \subseteq \mathbb{Z}_p$ e il crittosistema risulti omomorfo rispetto all'insieme $\mathcal{F}'$. Diciamo che SWHE e MHE sono* ***chimericamente compatibili*** *se esiste un insieme di polinomi $\mathcal{P} = \{P_j\}$ e delle limitazioni D e B tali che valgano le seguenti condizioni:*

- *per ogni testo cifrato c da SWHE, la funzione di decodifica* $\mathsf{SWHE.Dec}(\mathsf{sk}, c)$ *può essere valutata da un circuito $\mathcal{P}$-ristretto su $\mathbb{Z}_p$ con $\mathcal{P}$-grado D. Inoltre un descrizione esplicita di questo circuito può essere calcolata dato c.*
- *per ogni chiave segreta* sk *di SWHE e ogni polinomio $P_j \in \mathcal{P}$ abbiamo $P_j(\mathsf{sk}) \in \mathcal{M}$, lo spazio dei messaggi in chiaro di MHE.*
- *$\mathcal{F}'$ include tutti i prodotti di D (o meno) variabili.*
- *SWHE è in grado di valutare correttamente la funzione di decodifica di MHE,* $\mathsf{MHE.Dec}(\mathsf{sk}, c)$*, seguita da un polinomio quadratico (con un numero di termini polinomiale rispetto a un parametro di sicurezza) su $\mathbb{Z}_p$: la funzione* $\mathsf{MHE.Dec}(\mathsf{sk}, c)$ *aumentata sta in $\mathcal{F}$.*

A fronte di queste ultime strade osserviamo quanto la ricerca di un crittosistema pienamente omomorfo efficiente sia assiduamente perseguita. Continuano, infatti, ad esserci nuovi risultati volti a ottimizzare le procedure e a rendere realizzabile nella pratica la piena omomorfia. Citiamo inoltre un risultato recentissimo: nel mese di aprile 2013 è stata rilasciata una libreria che implementa la crittografia omomorfa, in particolare il crittosistema BGV, che si sta rivelando molto veloce nella valutazione dei circuiti sui testi cifrati. La libreria si chiama HElib, è scritta in C++ e usa la libreria matematica NTL. È disponibile tramite GitHub, una open source

community, all'indirizzo `https://github.com/shaih/HElib`.

Possiamo sperare che il traguardo di una efficiente implementazione della piena omomorfia venga presto raggiunto, perché da essa le applicazioni seguirebbero numerose ([22]) nell'ambito del cloud-computing, del voto elettronico, del multi-party computation ed in generale nei casi in cui vogliamo delegare ad altri il calcolo su nostri dati codificati (ad esempio: una ricerca su dati personali o un'analisi dei dati in ambito medico o finanziario).

Appendice A

Valutare $\zeta(2)$

Nel primo capitolo abbiamo utilizzato la funzione zeta di Riemann $\zeta(s)$, con $s = 2$, al fine di calcolare la probabilità che, scelti casualmente due interi, essi risultino coprimi.
Trattandosi di una funzione che riveste una fondamentale importanza in molti ambiti della matematica, dalla teoria dei numeri alla teoria della probabilità, dedichiamo una sezione alla sua definizione e alla dimostrazione del valore di $\zeta(2) = \pi^2/6$.

Definizione A.1. *La funzione zeta di Riemann è definita come la serie di Dirichlet*

$$\zeta(s) = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \ldots = \sum_{n=1}^{\infty} \frac{1}{n^s},$$

per ogni numero complesso s di parte reale $Re(s)$ maggiore di 1.

La restrizione ai complessi con parte reale maggiore di 1 è necessaria affinché la serie risulti convergente, tuttavia la funzione si può prolungare analiticamente a una funzione olomorfa su tutto il piano complesso ad eccezione di 1, dove ha un polo semplice. L'importanza della funzione zeta dipende da una fondamentale identità scoperta da Eulero, che esprime la funzione come prodotto esteso solo ai numeri primi.

Teorema A.2. *La funzione zeta di Riemann $\zeta(s)$è esprimibile attraverso il seguente prodotto svolto su tutti i primi p*

$$\zeta(s) = \prod_{p \text{ primo}} \left(1 - \frac{1}{p^s}\right)^{-1}, \tag{A.1}$$

con $s \in \mathbb{C}$, $Re(s) > 1$.

Dimostrazione. Cominciamo col prodotto

$$\prod_{p} \left(1 - \frac{1}{p^s}\right)^{-1} = \left(1 - \frac{1}{2^s}\right)^{-1} \left(1 - \frac{1}{3^s}\right)^{-1} \ldots$$

e scriviamo ogni fattore nel seguente modo

$$\left(1-\frac{1}{p^s}\right)^{-1}=\frac{1}{1-\frac{1}{p^s}}=1+\frac{1}{p^s}+\frac{1}{p^{2s}}+\frac{1}{p^{3s}}+\dots$$

Prendiamo il prodotto sui primi minori o uguali di un fissato N e denotiamo con $\overline{p}$ il più grande numero primo che soddisfa la condizione $\overline{p}\leq N$, abbiamo

$$\begin{aligned}P(N)=\prod_{p\leq N}\left(1+\frac{1}{p^s}+\frac{1}{p^{2s}}+\dots\right) &= \left(1+\frac{1}{p^s}+\frac{1}{p^{2s}}+\dots\right)\dots\left(1+\frac{1}{\overline{p}^s}+\frac{1}{\overline{p}^{2s}}+\dots\right)\\ &= 1+\frac{1}{2^s}+\frac{1}{3^s}+\frac{1}{2^{2s}}+\frac{1}{5^s}+\frac{1}{2^s3^s}+\frac{1}{7^s}+\dots\end{aligned}$$

Per il teorema fondamentale dell'aritmetica $P(N)=\sum_{n\in A}1/n^s$, dove A è l'insieme di tutti gli n aventi fattori primi minori o uguali a N. Abbiamo

$$\sum_{n=1}^{\infty}\frac{1}{n^s}-P(N)=\sum_{n\in B}\frac{1}{n^s},$$

dove B è l'insieme dei naturali n aventi almeno un fattore primo maggiore di N. Pertanto

$$\left|\sum_{n=1}^{\infty}\frac{1}{n^s}-P(N)\right|\leq\sum_{n\in B}\left|\frac{1}{n^s}\right|\leq\sum_{n>N}\left|\frac{1}{n^s}\right|.$$

Se $N\to\infty$ l'ultima somma sulla destra tende a 0, dato che, per $Re(s)>1$, $\sum_{n>N}|1/n^s|$ è convergente. Quindi $P(N)\to\zeta(s)$ quando $N\to\infty$. □

Ora dimostriamo il risultato utilizzato in precedenza:

$$\zeta(2)=\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{\pi^2}{6}. \tag{A.2}$$

Dimostrazione. Usiamo il prodotto infinito per la funzione seno:

$$\begin{aligned}\sin\pi x &= \pi x\prod_{n=1}^{\infty}\left(1-\frac{x^2}{n^2}\right)\\ &= \pi x(1-x^2)\left(1-\frac{x^2}{2^2}\right)\left(1-\frac{x^2}{3^2}\right)\dots\\ &= \pi(x-x^3)\left(1-\frac{x^2}{2^2}\right)\left(1-\frac{x^2}{3^2}\right)\dots\\ &= \pi\left(x-\frac{x^3}{2^2}-x^3-\frac{x^5}{2^2}\right)\left(1-\frac{x^2}{3^2}\right)\dots\\ &= \pi\left(x-x^3-\frac{x^3}{2^2}-\frac{x^3}{3^2}-\frac{x^5}{2^2}+\dots+\frac{x^7}{6^2}\right)\left(1-\frac{x^2}{4^2}\right)\dots\\ &= \pi\left(x-x^3-\frac{x^3}{2^2}-\frac{x^3}{3^2}-\frac{x^3}{4^2}-\frac{x^5}{2^2}+\dots+\frac{x^9}{24^2}\right)\left(1-\frac{x^2}{5^2}\right)\dots\end{aligned}$$

Confrontando coefficienti di x^3 nel prodotto infinito con la serie di MacLaurin

$$\begin{aligned}\sin \pi x &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}(\pi x)^{2n+1} \\ &= \pi x - \frac{(\pi x)^3}{6} + \dots\end{aligned}$$

si ottiene subito che $\zeta(2) = \pi^2/6$.

Equivalentemente possiamo dimostrare un risultato più generale per $\zeta(2k)$, con k un numero naturale. Per $|z| < \pi$ abbiamo

$$\begin{aligned}z \cot z = z\frac{\cos z}{\sin z} &= 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2 - z^2} \\ &= 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2} \frac{1}{1 - (\frac{z}{n\pi})^2} \\ &= 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2} \sum_{k=0}^{\infty} \left(\frac{z}{n\pi}\right)^{2k} \\ &= 1 - 2\sum_{k=0}^{\infty} \left(\sum_{n=1}^{\infty} \frac{1}{n^{2k+2}}\right) \frac{z^{2k+2}}{\pi^{2k+2}} \\ &= 1 - 2\sum_{k=1}^{\infty} \left(\sum_{n=1}^{\infty} \frac{1}{n^{2k}}\right) \frac{z^{2k}}{\pi^{2k}}\end{aligned}$$

Un'altra formula per $z \cot z$:

$$z \cot z = z\frac{\cos z}{\sin z} = iz + \frac{2iz}{e^{2iz} - 1} = 1 + \sum_{k+2}^{\infty} B_k \frac{(2iz)^k}{k!},$$

dove B_k sono numeri di Bernoulli. Confrontando i coefficienti delle potenze di z^k ottenuti nei due modi, otteniamo

$$\zeta(2k) = (-1)^{k+1} \frac{(2\pi)^{2k}}{2(2k)!} B_{2k},$$

da cui, per $k = 1$, $\zeta(2) = \pi^2/6$. □

Diamo un'altra dimostrazione, presa dall'articolo di Tom M. Apostol [4].

Dimostrazione. Notiamo che

$$\frac{1}{n^2} = \int_0^1 \int_0^1 x^{n-1} y^{n-1} dx\, dy$$

e dal teorema della convergenza monotona abbiamo

$$\sum_{n=1}^{\infty}\frac{1}{n^2}=\int_0^1\int_0^1\left(\sum_{n=1}^{\infty}(xy)^{n-1}\right)dx\;dy=\int_0^1\int_0^1\frac{dx\;dy}{1-xy}.$$

Operiamo un cambio di variabili $(u,v)=((x+y)/2,(y-x)/2)$, da cui $(x,y)=(u-v,u+v)$. Il nostro integrale diventa

$$\zeta(2)=2\int\int_S\frac{du\;dv}{1-u^2+v^2},$$

dove S è il quadrato di vertici $(0,0)$, $(1/2,-1/2)$, $(1,0)$ e $(1/2,1/2)$. Sfruttando la simmetria del quadrato, possiamo scrivere

$$\begin{aligned}\zeta(2) &= 4\int_0^{1/2}\int_0^{u}\frac{dv\;du}{1-u^2+v^2}+4\int_{1/2}^{1}\int_0^{1-u}\frac{dv\;du}{1-u^2+v^2}\\ &= 4\int_0^{1/2}\frac{1}{\sqrt{1-u^2}}\tan^{-1}\left(\frac{u}{\sqrt{1-u^2}}\right)du+4\int_{1/2}^{1}\frac{1}{\sqrt{1-u^2}}\tan^{-1}\left(\frac{1-u}{\sqrt{1-u^2}}\right)du\end{aligned}$$

Dato che $\tan^{-1}\left(\frac{u}{\sqrt{1-u^2}}\right)=\sin^{-1}u$ e, se $\theta=\tan^{-1}\left(\frac{1-u}{\sqrt{1-u^2}}\right)$, allora $\tan^2\theta=\frac{1-u}{\sqrt{1+u}}$ e $\sec^2\theta=\frac{2}{\sqrt{1+u}}$. Segue che $u=2\cos^2\theta-1=\cos 2\theta$ e così $\theta=\frac{1}{2}\cos^{-1}u=\frac{\pi}{4}-\frac{1}{2}\sin^{-1}u$. Quindi

$$\begin{aligned}\zeta(2) &= 4\int_0^{1/2}\int_0^{u}\frac{\sin^{-1}u}{\sqrt{1-u^2}}du+4\int_{1/2}^{1}\frac{1}{\sqrt{1-u^2}}\left(\frac{\pi}{4}-\frac{\sin^{-1}u}{2}\right)du\\ &= [2(\sin^{-1}u)^2]_0^{1/2}+[\pi\sin^{-1}u-(\sin^{-1}u)^2]_{1/2}^{1}\\ &= \frac{\pi^2}{18}+\frac{\pi^2}{2}-\frac{\pi^2}{4}-\frac{\pi^2}{6}+\frac{\pi^2}{36}\\ &= \frac{\pi^2}{6}\end{aligned}$$

come volevamo. □

Un'utile applicazione della funzione zeta di Riemann consiste nel calcolo della probabilità che, scelti casualmente due interi, essi risultino coprimi.

Metodo 1. Per ogni $n\in\mathbb{N}$ abbiamo

$$\gcd(x,y)=n\Leftrightarrow\begin{cases}x\equiv 0\bmod n\\ y\equiv 0\bmod n\\ \gcd(x/n,y/n)=1\end{cases}$$

Le condizioni $x\equiv 0\bmod n$ e $y\equiv 0\bmod n$ sono soddisfatte con probabilità $1/n$; dato che x e y sono scelti indipendentemente, queste due condizioni sono simultaneamente valide con

probabilità $1/n^2$ e denominiamo P la probabilità che i due interi x/n e y/n siano coprimi. Segue che le tre condizioni valgono contemporaneamente con probabilità P/n^2, ovvero $Pr(\gcd(x,y) = n) = P/n^2$, e, prendendo la somma su n, si ha

$$1 = \sum_{n=1}^{\infty} Pr(\gcd(x,y) = n) = \sum_{n=1}^{\infty} \frac{P}{n^2} = P \sum_{n=1}^{\infty} \frac{1}{n^2} = P\zeta(2);$$

quindi $P = 1/\zeta(2)$.

Metodo 2. Abbiamo

$$\gcd(x,y) = 1 \Leftrightarrow (x \not\equiv 0 \bmod p \text{ o } y \not\equiv 0 \bmod p)$$

per ogni primo p. La probabilità che la condizione $x \equiv 0 \bmod p$ sia soddisfatta è di $1/p$, da cui la probabilità che entrambe le condizioni siano verificate $x \equiv 0 \equiv y \bmod p$ vale $1/p^2$, e quindi abbiamo $x \not\equiv 0 \bmod p$ o $y \not\equiv 0 \bmod p$ con probabilità $1 - 1/p^2$. Dato che le congruenze modulo primi distinti sono statisticamente indipendenti, la probabilità su tutti i primi p vale

$$P = \prod_{p} \left(1 - \frac{1}{p^2}\right) = \frac{1}{\zeta(2)}.$$

Appendice B

Classi P e NP

Una macchina di Turing deterministica (DTM) consiste di:

- un alfabeto finito Σ contenente un simbolo speciale $*$ che corrisponde ad una casella vuota;
- un nastro diviso in celle, una delle quali è la casella di partenza. Ogni cella contiene un carattere dell'alfabeto Σ. Tutte tranne un numero finito di caselle contengono il simbolo $*$;
- una testina di lettura e scrittura che esamina una singola cella alla volta e può muoversi verso sinistra ($\leftarrow$) o verso destra ($\rightarrow$);
- un'unità di controllo e un insieme finito di stati Γ che include un particolare stato di partenza γ_0, e un sottoinsieme di stati finali.

Lo svolgimento delle azioni di una DTM è controllato da una funzione di transizione

$$\delta : \ \Gamma \times \Sigma \rightarrow \Gamma \times \Sigma \times \{\leftarrow, \rightarrow\}.$$

Inizialmente l'unità di controllo è nello stato di partenza γ_0 e la testina lettura-scrittura analizza la cella iniziale, la funzione di transizione dice alla macchina cosa fare come prossima mossa in base al contenuto della casella e allo stato corrente dell'unità di controllo. Per esempio se l'unità di controllo è nello stato γ_{cor} e la cella contiene il simbolo σ_{cor} allora il valore $\delta(\gamma_{cor}, \sigma_{cor})$ dice alla macchina tre cose:

- il nuovo stato per l'unità di controllo (se corrisponde ad uno stato finale, l'esecuzione terminerà),

- il simbolo da scrivere nella casella corrente,
- se muovere la testina alla cella di destra o di sinistra.

Usiamo Σ_0 per denotare l'alfabeto senza il simbolo vuoto, $\Sigma \setminus \{*\}$, e Σ_0^* per indicare la collezione di tutte le stringhe finite a valori in Σ_0. Per $x \in \Sigma_0^*$, utilizziamo la notazione $|x|$ per indicare la lunghezza di x. Il calcolo eseguito da una DTM su un input $x \in \Sigma_0^*$ è il risultato ottenuto applicando la funzione di transizione ripetutamente a partire dalla stringa x posta sul nastro, mentre una singola applicazione della funzione di transizione è chiamata step. Una configurazione di una DTM è una descrizione completa della macchina in un particolare punto del calcolo: il contenuto del nastro, la posizione della testina e lo stato corrente dell'unità di controllo.
Se una DTM si ferma durante un calcolo su un input $x \in \Sigma_0^*$, allora il suo tempo d'esecuzione $t_M(x)$ su x è il numero di step che la macchina svolge durante il calcolo. La complessità T_M di una macchina di Turing M è definita come la funzione $\mathbb{N} \to \mathbb{N}$ data da

$$T_M(n) = \max\{t \ : \ \exists x \in \Sigma_0^n \text{ tale che } t_M(x) = t\},$$

dove Σ_0^n è l'insieme di stringhe prese da Σ_0^* di lunghezza n.
Una particolare categoria di problemi riveste un ruolo importante nella teoria della complessità, quelli per cui l'output è vero o falso, questi problemi si dicono decisionali; ad esempio dire se un numero è primo è un problema decisionale.
Una macchina di Turing viene detta accettore se ha esattamente due stati finali: γ_V e γ_F, corrispondenti, rispettivamente, alla condizione di verità e falsità. Un input $x \in \Sigma_0^*$ è accettato da un accettore se esso si ferma nello stato γ_V, mentre è rifiutato se lo stato finale è γ_F. Un sottoinsieme di stringhe $L \subseteq \Sigma_0^*$ è chiamato linguaggio; se M è un accettore allora definiamo un linguaggio accettato da M come $L(M) = \{x \in \Sigma_0^* \ : \ x \text{ accettati da } M\}$. Se $L = L(M)$ e M si ferma su tutti gli input $x \in \Sigma_0^*$, allora diciamo che M decide L.
L'obiettivo della teoria della complessità è capire la difficoltà dei problemi computazionali, suddividendo collezioni di linguaggi in base alla complessità con cui una DTM li decide. Tali collezioni di linguaggi per cui la complessità della DTM è la medesima si chiama classe di complessità. Una classe fondamentale è la classe dei linguaggi decidibili in un tempo polinomiale, detta P:

$$\begin{aligned} \mathrm{P} \ = \ \{&L \subseteq \Sigma_0^* \ : \ \text{esistono una DTM } M \text{ che decide } L \text{ e un polinomio } p(n) \\ &\text{tali che } T_M(n) \leq p(n) \text{ per ogni } n \geq 1\}. \end{aligned}$$

Se Π è un problema decisionale per il quale $L_\Pi \in \mathrm{P}$, diciamo che esiste un algoritmo con tempo polinomiale per Π. Osserviamo che le classi di complessità C contengono linguaggi e non problemi, ma spesso si abusa della notazione e si indica $\Pi \in \mathrm{C}$. Possiamo comunque estendere la definizione alle funzioni; per esempio, consideriamo la funzione $\mathsf{factor}(n) : \mathbb{N} \to \mathbb{N}$,

$$\mathsf{factor}(n) = \begin{cases} d & \text{il più piccolo fattore non banale di } n, \text{ se esiste,} \\ n & \text{altrimenti.} \end{cases}$$

Un algoritmo efficiente che calcoli $\mathsf{factor}(n)$ violerebbe gran parte dei crittosistemi più comuni; per questa ragione, determinare la complessità di una tale funzione risulta un problema importante. La classe di funzioni trattabili FP, analoga alla classe P definita in precedenza, può essere così descritta

$$\begin{aligned} \mathrm{FP} \;=\; \{f : \Sigma_0^* \to \Sigma_0^* \;:\; &\text{esistono una DTM } M \text{ che calcoli } f \text{ e un polinomio } p(n) \\ &\text{tali che } T_M(n) \leq p(n) \text{ per ogni } n \geq 1\}. \end{aligned}$$

Se $f \in \mathrm{FP}$, diciamo che f è calcolabile in un tempo polinomiale.

Fino a questo punto abbiamo considerato le esecuzioni solo da un punto di vista temporale, ma un'altra limitazione delle capacità di calcolo è costituita dallo spazio e cerchiamo di definire la complessità spaziale di una DTM. Se una macchina di Turing si ferma su un input $x \in \Sigma_0^*$, allora lo spazio $s_M(x)$ usato su x è il numero di celle distinte esaminate dalla testina di lettura-scrittura durante il calcolo. Se M è una DTM che si ferma per ogni input $x \in \Sigma_0^*$, allora la complessità spaziale di M è la funzione $S_M : \mathbb{N} \to \mathbb{N}$ definita da

$$S_M(n) = \max\{s \;:\; \text{esiste } x \in \Sigma_0^* \text{ tale che } s_M x = s\}.$$

La classe più importante è quella dei linguaggi che possono essere decisi in uno spazio polinomiale

$$\begin{aligned} \mathrm{PSPACE} \;=\; \{L \subseteq \Sigma_0^* \;:\; &\text{esistono una DTM } M \text{ che decide } L \text{ e un polinomio } p(n) \\ &\text{tali che } S_M(n) \leq p(n) \text{ per ogni } n \geq 1\}. \end{aligned}$$

Grazie alla proposizione che segue, abbiamo un legame tra le complessità in termini di tempo e spazio, precisamente $\mathrm{P} \subseteq \mathrm{PSPACE}$.

Proposizione B.1. *Se un linguaggio L può essere deciso in un tempo $f(n)$ allora L può essere deciso in uno spazio $f(n)$.*

Dimostrazione. Il numero di celle esaminate dalla testina di lettura-scrittura di una DTM non può essere superiore al numero di step che richiede il calcolo. □

Consideriamo un esempio di problema decisionale che ci sarà utile per definire un'altra classe di complessità.
Innanzitutto chiamiamo una funzione f booleana se $f : \{0,1\}^n \to \{0,1\}$, in cui interpretiamo 1 come vero e 0 come falso. Le funzioni Booleane di base sono la negazione NOT, la congiunzione AND e la disgiunzione OR. Se x è una variabile booleana allora la sua negazione è

$$\overline{x} = \begin{cases} 1 & \text{se } x = 0 \\ 0 & \text{altrimenti.} \end{cases}$$

Detto che un letterale è una variabile booleana o la sua negazione, definiamo la congiunzione di una collezione di letterali $x_1, \ldots, x_n$ come segue

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n = \begin{cases} 1 & \text{se } x_i = 1 \ \forall i \in \{1, \ldots, n\} \\ 0 & \text{altrimenti.} \end{cases}$$

Infine la disgiunzione di una collezione di letterali $x_1, \ldots, x_n$ è

$$x_1 \vee x_2 \vee \ldots \vee x_n = \begin{cases} 1 & \text{se } x_i = 1 \ \exists i \in \{1, \ldots, n\} \\ 0 & \text{altrimenti.} \end{cases}$$

Una funzione booleana è detta in forma normale congiuntiva (conjunctive normal form, CNF) se è scritta nella seguente forma

$$f(x_1, \ldots, x_n) = \bigwedge_{k=1}^{m} C_k,$$

per un certo valore intero m e con C_k è una somma logica (disgiunzione) di letterali. Si definisce assegnazione di verità per una collezione di variabili booleane $\mathbf{x} = \{x_1, \ldots, x_n\}$, l'associazione di un valore in $\{0,1\}$ a ogni x_i . Un'assegnazione di verità si dice soddisfacente quando $f(\mathbf{x})$ risulta uguale a 1. La formula si dice soddisfacibile se le variabili possono essere assegnate in modo che la formula assuma il valore di verità 1.
La soddisfacibilità booleana, conosciuta anche come SAT, è il seguente problema decisionale:
SAT
Input: una funzione booleana $f(x_1, \ldots, x_n) = \bigwedge_{k=1}^{m} C_k$, in CNF;
domanda: f è soddisfacibile?
Consideriamo questo banale algoritmo risolutivo di SAT:

- Input: una funzione booleana $f(x_1, \ldots, x_n) = \bigwedge_{k=1}^{m} C_k$, in CNF;

- si prenda una possibile assegnazione di verità $\mathbf{x} \in \{0,1\}^n$, se risulta $f(\mathbf{x}) = 1$ allora l'algoritmo si arresta e l'output è vero, altrimenti si prosegue scegliendo un'altra assegnazione in cui valutare f.

- Output: vero se f è soddisfacibile e falso altrimenti.

L'algoritmo risulta decisamente poco pratico se f non è soddisfacibile, perché in tal caso dovremmo provare 2^n assegnazioni di verità prima di fermarci e avremmo un tempo d'esecuzione esponenziale. Ci troviamo nella situazione in cui valutare f in una fissata assegnazione di verità è facile e veloce, mentre risolvere il problema provando tutte le assegnazioni è molto lento. Se noi fossimo in possesso di una assegnazione soddisfacibile, questa certificherebbe la soddisfacibilità della funzione, proprio per questo motivo chiamiamo questa assegnazione di verità soddisfacibile *certificato* (succinct certificate).
In generale, quando un problema decisionale Π ha un certificato che può essere usato per valutare la sua verità in tempo polinomiale, diciamo che il linguaggio associato L_Π è accettato in tempo polinomiale non-deterministico o, equivalentemente, che L_Π appartiene alla classe di complessità NP. Possiamo formalizzare quanto detto come segue.
Per $x, y \in \Sigma_0^*$, denotiamo con $x * y$ la stringa che consiste di x seguita da una cella vuota e in coda da y. Diciamo che un linguaggio $L \subseteq \Sigma_0^*$ appartiene a NP se esistono una macchina di Turing deterministica M e un polinomio $p(n)$ tali che $T_M(n) \leq p(n)$ e su un ogni input $x \in \Sigma_0^*$:

- se $x \in L$ allora esiste un certificato $y \in \Sigma_0^*$ tale che $|y| \leq p(|x|)$ e M accetta in input la stringa $x * y$;

- se $x \notin L$ allora, per ogni stringa $y \in \Sigma_0^*$, M rifiuta la stringa $x * y$

In altre parole, un linguaggio L appartiene a NP se esiste un algoritmo con tempo polinomiale che, quando gli viene dato in input $x \in L$ insieme ad un certificato di lunghezza polinomiale y, accetta x, ma quando gli viene dato un input $x \notin L$, lo rifiuta sempre, a prescindere dal certificato y.

Sorge spontaneo chiedersi che legame ci sia tra la classe P e quella NP ed è immediato vedere che P $\subseteq$ NP.

Proposizione B.2. *La classe di complessità P è contenuta in NP: P $\subseteq$ NP.*

Dimostrazione. Se $L \in P$ allora esiste una DTM con tempo d'esecuzione polinomiale che decide L, quindi non abbiamo bisogno di un certificato che verifichi che un particolare input $x \in \Sigma_0^*$ appartenga a L. L'algoritmo, allora, prende un input $x \in \Sigma_0^*$ e decide direttamente se $x \in L$ o meno, in tempo polinomiale. □

È comunemente ritenuto che ci siano problemi di classe NP che non siano risolvibili in un tempo d'esecuzione polinomiale da una macchina di Turing deterministica, ovvero P $\neq$ NP, tuttavia esso rimane un importante problema aperto, uno dei cosiddetti "problemi del Millennio" scelti dal Clay Institute.

Teorema B.3. *NP $\subseteq$ PSPACE.*

Ci sono molte situazioni in cui la capacità di risolvere un problema Π_1 permette di risolvere un secondo problema Π_2, perché quest'ultimo è riconducibile al primo.
Date le stringhe $A, B \in \Sigma_0^*$, diciamo che $f : \Sigma_0^* \to \Sigma_0^*$ è una riduzione da A a B se soddisfa la seguente doppia implicazione: $x \in A \Leftrightarrow f(x) \in B$; inoltre, se $f \in FP$, chiamiamo f una riduzione in tempo polinomiale. Se una tale funzione esiste diciamo che A è polinomialmente riducibile a B e scriviamo $A \leq_m B$.
Avendo introdotto la nozione di un linguaggio difficile quanto un altro linguaggio, sorge spontaneo chiedersi se esistono esempi di linguaggi che appartengono a NP e che sono almeno difficili quanto un altro linguaggio in NP, ossia se NP contiene i linguaggi più difficili. In accordo con quanto appena detto, definiamo un linguaggio NP-completo se

- il linguaggio $L \in$ NP,
- se $A \in$ NP allora $A \leq_m L$.

Proposizione B.4. *Se un linguaggio NP-completo appartiene a P allora P = NP.*

Dimostrazione. Dato che P $\subseteq$ NP, è sufficiente mostrare che con queste assunzioni vale anche l'inclusione inversa, NP $\subseteq$ P. Chiamiamo L il linguaggio NP-completo appartenente a P dell'ipotesi. Consideriamo una stringa A di un linguaggio contenuto in NP, allora $A \leq_m L$ e, dato che $L \in$ P, si ha $A \in$ P. Quindi NP $\subseteq$ P, come volevamo. □

Nella dimostrazione precedente abbiamo usato la seguente proprietà delle riduzioni polinomiali:

Lemma B.5. *Siano $A, B \in \Sigma_0^*$ tali che $A \leq_m B$ e $B \in P$, allora $A \in P$.*

Dimostrazione. Se $A \leq_m B$ e $B \in$ P allora esistono due macchine di Turing deterministiche, M e N, con le seguenti proprietà:

1. M calcola una funzione $f : \Sigma_0^* \to \Sigma_0^*$ che soddisfa $x \in A \Leftrightarrow f(x) \in B$;
2. esiste un polinomio $p(n)$ tale che $T_M(n) \leq p(n)$;
3. N decide il linguaggio B;
4. esiste un polinomio $q(n)$ tale che $T_N(n) \leq q(n)$.

Costruiamo una DTM che decida A in un tempo d'esecuzione polinomiale. Dato $x \in \Sigma_0^n$, diamo x come input a M per ottenere $f(x)$. Passiamo poi $f(x)$ a N che lo accetterà o lo rigetterà. Dato che M esegue una riduzione da A a B e N decide il linguaggio B, la nostra nuova macchina sarà in grado di decidere A. Per vedere che lo farà in un tempo polinomiale, notiamo che il tempo necessario a M per calcolare $f(x)$ è al massimo $p(n)$. Inoltre $T_M(n) \leq p(n)$ implica che $|f(x)| \leq p(n) + n$. Questo perché la macchina M comincia con una stringa di lunghezza n sul nastro e si ferma dopo al più $p(n)$ step, così quando si ferma non può avere più di $p(n) + n$ celle non vuote. Allora il tempo impiegato da N per fornire $f(x)$ è al massimo $q(p(n) + n)$. Quindi il tempo d'esecuzione totale della nostra macchina è al massimo $p(n) + q(p(n) + n)$, che è ancora polinomiale in n, la dimensione dell'input, pertanto $A \in$ P. □

Una funzione f è detta Turing-riducibile a una funzione g, $f \leq_T g$, se un algoritmo eseguibile da una macchina di Turing in un tempo polinomiale permette il calcolo di f sempre in tempo polinomiale. Diciamo che una funzione f è NP-hard se esiste un linguaggio NP-completo L tale che $L \leq_T f$, dove identifichiamo L con f_L, la funzione corrispondente al linguaggio

$$f_L : \Sigma \to \{0, 1\}, \quad f_L = \begin{cases} 1 & \text{se } x \in L \\ 0 & \text{se } x \notin L \end{cases}.$$

Quindi una funzione NP-hard è difficile almeno quanto ogni altro linguaggio in NP, nel senso che un algoritmo con tempo d'esecuzione polinomiale che sia in grado di calcolare una tale funzione fornirebbe un algoritmo capace di calcolare in tempo polinomiale ogni linguaggio in NP.

Un'altra tipologia di macchina di Turing, oltre a quella deterministica, è quella non deterministica o NTM. Possiamo definirla in modo simile alla DTM, eccetto che al posto della

funzione di transizione la NTM ha una relazione di transizione, in modo che in ogni momento dell'esecuzione essa possa scegliere tra diverse azioni possibili, in modo non deterministico appunto. Ricordiamo che la funzione di transizione $\delta : \Gamma \times \Sigma \to \Gamma \times \Sigma \times \{\leftarrow, \rightarrow\}$ di una DTM restituisce un'unica possibile combinazione di $\Gamma \times \Sigma \times \{\leftarrow, \rightarrow\}$. Per una NTM, invece, abbiamo una relazione di transizione $\Delta \subseteq (\Gamma \times \Sigma) \times (\Gamma \times \Sigma \times \{\leftarrow, \rightarrow\})$.
Più precisamente, sia N una NTM, se la macchina è nello stato γ_c e il contenuto della cella è σ_c, allora al prossimo step N sceglie una possibile azione in modo non deterministico dall'insieme

$$\Delta(\gamma_c, \sigma_c) = \{(\gamma_n, \sigma_n, m_n) \ : \ ((\gamma_c, \sigma_c), (\gamma_n, \sigma_n, m_n)) \in \Delta\}.$$

Data una stringa $x \in \Sigma_0^*$, una computazione sull'input x è il risultato dell'applicazione ripetuta della relazione di transizione, che termina se viene raggiunto lo stato finale. Notiamo che per ogni dato input x ci saranno più computazioni possibili. Diciamo, poi, che un input $x \in \Sigma_0^*$ è accettato da una NTM se esiste una computazione su x che si fermi allo stato γ_T. Per una NTM, N, definiamo un linguaggio accettato da N come

$$L(N) = \{x \in \Sigma_0^* \ : \ x \text{ è accettato da } N\}.$$

Analogamente a quanto fatto per una DTM, assumiamo che uno step in una computazione sia il risultato di una singola applicazione della relazione di transizione. Per $x \in L(N)$ definiamo il tempo impiegato per accettare x come il numero di step nella computazione più corta che accetta x:

$$t_N(x) = \min\{t \ : \ \text{esiste una computazione eseguita da } N \text{ che accetta } x \text{ e che si ferma dopo } t \text{ step}\}.$$

La complessità (temporale) di N è definita

$$T_N(n) = \max\{t \ : \ \exists x \in L(N) \text{ tale che } |x| = n \text{ e } t_N(x) = t\}.$$

Si può anche definire la classe NP come la classe dei linguaggi accettati da una macchina di Turing non deterministica in tempo polinomiale.
Riportiamo una rappresentazione delle classi di complessità che possa chiarire quanto definito in questa appendice.

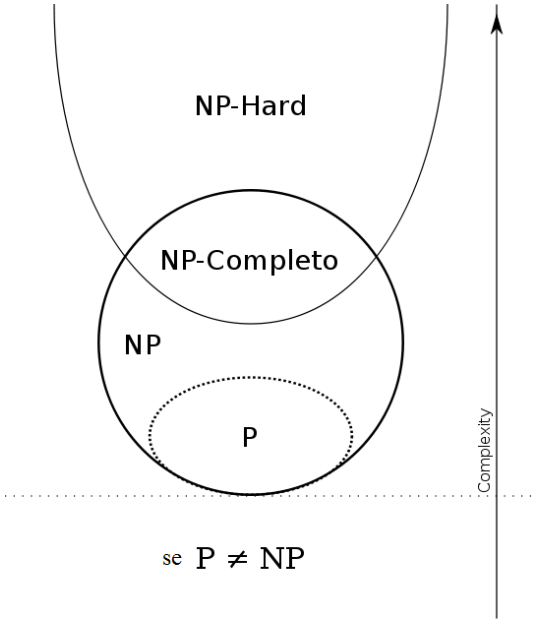

Figura B.1: Le classi di complessità

Appendice C

Implementazione

Per rendere attuabile nella pratica un crittosistema omomorfo ci sono varie possibilità. Trattandosi di un argomento in piena fase di sviluppo, il materiale che è possibile reperire segue strade differenti, proprio per cercare di capire quale può costituire la migliore via di realizzazione. È pertanto utile dare un po' di chiarezza su quanto abbiamo a disposizione attualmente, secondo le versioni più recenti. Qui proponiamo l'implementazione del crittosistema BGV [5] tramite la libreria HElib, di cui abbiamo dato un accenno nel capitolo precedente, e della versione basata sull'aritmetica modulare realizzata da Coron, Mandal, Naccache e Tibouchi in [8] e [9].

Implementazione del crittosistema BGV

Per prima cosa precisiamo che è necessario avere Linux come sistema operativo (purtroppo Windows non è ancora in grado di supportare la libreria HElib), ma questo ha il vantaggio di rendere i passaggi di installazione molto rapidi, basta solo assicurarsi di aver installato il compilatore per C++ (per fare ciò bisogna solo scrivere nella shell `sudo apt-get install build-essential`).

Abbiamo bisogno dell'ultima versione della libreria NTL (attualmente è la 6.0.0), la cui creazione è dovuta a Victor Shoup e con la quale è possibile lavorare nell'ambito della Teoria dei Numeri. La libreria è scaricabile gratuitamente al seguente indirizzo `http://www.shoup.net/ntl/download.html`, dove si trovano le versioni per qualsiasi piattaforma, 32 o 64 bit con sistema operativo Unix (e suoi derivati) o Windows. Una volta scaricata, basta dare i seguenti comandi dal terminale:

```
gunzip ntl-xxx.tar.gz
tar xf ntl-xxx.tar
cd ntl-xxx/src
```

```
./configure
make
make check
sudo make install,
```

dove `xxx` indica il numero della versione che si sta installando e `sudo` all'inizio dell'ultima riga è necessario per eseguire il comando con i permessi da amministratore.
Dopodiché si passa a scaricare la libreria HElib, che si trova all'indirizzo `https://github.com/shaih/HElib`. Una volta estratta dall'archivio, tramite la shell si cambia directory per mettersi in HElib-master/src e da qui si esegue semplicemente il comando `make`. All'interno della cartella src di HElib si trovano dei test; per iniziare a comprendere i meccanismi dell'implementazione è utile eseguire Test_General.cpp. Si compila il file col comando `g++ -g -O2 -Wfatal-errors -Wshadow -Wall -I/home/.../ntl-6.0.0/include -o Test_General Test_General.cpp fhe.a -lntl -lm`[1] e poi per eseguirlo basta scrivere `./Test_General`.
I risultati che si ottengono sono i parametri utilizzati dal crittosistema e il rapporto tra i tempi di esecuzione di ogni funzione e il numero di chiamate alla funzione stessa; questo si può capire compilando ed eseguendo il file timing.cpp.
Infine in HElib/doc/latex si trova il file refman, dove sono riportate sinteticamente le spiegazioni delle funzioni utilizzate.
All'indirizzo `http://code.google.com/p/thep/downloads/list` si può trovare un'applicazione della libreria HElib: HEcalc. Si tratta di un calcolatore in notazione polacca inversa (RPN)[2], grazie al quale poter ottenere il risultato di operazioni su valori da noi inseriti, passando attraverso la cifratura e la decifratura. In sintesi noi scegliamo i valori e le operazioni, ad esempio 3 4 $*$, il programma codifica i dati, svolge la moltiplicazione e decodifica il risultato, fornendoci 12. Per poter vedere questo basta modificare i file che troviamo nell'archivio scaricato. Innanzitutto si cambia la directory in Makefile con la propria directory in cui si trova HElib. Successivamente si aggiunge dopo la riga 62 del codice in HEcalc.cpp:

```
63  cout << " Ciphertext is:  " << c;
64  cout << endl;
```

[1] È possibile, ma non indispensabile, aggiungere la libreria di aritmetica multiprecisione GMP (`http://gmplib.org/`) e scrivere in tal caso `g++ -g -O2 -Wfatal-errors -Wshadow -Wall -I/home/.../ntl-6.0.0/include -o Test_General Test_General.cpp fhe.a -lntl -lgmp -lm`.

[2] La notazione polacca inversa è una sintassi utilizzata per scrivere espressioni matematiche, in cui gli operatori seguono gli operandi; per questo motivo viene detta anche notazione postfissa, a differenza della usuale notazione infissa. La RPN ha il vantaggio di permettere lo svolgimento di qualsiasi tipo di operazione senza l'utilizzo delle parentesi.

Dato che l'output sarà molto lungo, conviene eseguire HEcalc con il comando `./HEcalc|less` oppure modificare il codice per ottenere i testi cifrati su file di testo. Per fare ciò basta aggiungere la direttiva `#include <fstream>` e

```
ofstream risultati;
risultati.open("testHEcalc2.txt", ios::out);
```

per aprire il file in modalità scrittura. Poi basta inserire nel main loop

```
risultati << "Ciphertext evaluated is:  " << theStack.top();
risultati << endl;
```

subito dopo `evaluate(token[0])` per avere il risultato delle operazioni svolte sui testi cifrati e

```
risultati << "Ciphertext is:  " << c;
risultati << endl;
```

dopo la riga 62 (al posto di `cout`) per poter visionare le codifiche dei valori dati in input a schermo.

A titolo esemplificativo, se inseriamo la semplice operazione 3 4 $*$ otteniamo in totale 499 pagine di output.

Implementazione del crittosistema DGHV

Per implementare il crittosistema basato sull'aritmetica modulare del Capitolo 1, è necessario fare riferimento agli articoli [8] e [9], grazie ai quali è stata ridotta la dimensione della chiave pubblica, nonché i tempi di esecuzione della fase di pulitura dei testi cifrati. Per questa implementazione usiamo Sage, un software matematico per la teoria dei numeri, l'algebra e la geometria, scritto in Phyton, C++ e C. Anch'esso è open source (scaricabile dal sito `http://www.sagemath.org`) e integra sotto un'unica interfaccia diversi software, tra i quali SINGULAR (algebra commutativa), SciPy (matematica applicata), GAP (teoria dei gruppi), la libreria MWRANK (curve ellittiche), la libreria PARI/GP (teoria dei numeri), LinBox (algebra lineare) e la suddetta libreria NTL. Per installare Sage, basta estrarlo dall'archivio scaricato e, spostandosi nella directory risultante, eseguire il comando `./sage`. Successivamente andiamo all'indirizzo `https://github.com/coron/fhe`, dove troviamo l'implementazione del crittosistema. Per eseguirla basta scrivere:

```
sage:  load dghv.sage
sage:  testPkRecrypt().
```

La funzione testPkRecrypt() mostra la generazione delle chiavi, l'addizione e la moltiplicazione

di testi cifrati e la fase di refresh per quattro parametri diversi (toy, small, medium, large).
La differenza principale tra quanto descritto nel Capitolo 1 e l'implementazione proposta consiste nella riduzione della chiave pubblica. Nel crittosistema DGHV la dimensione dei testi cifrati deve essere grande (almeno 10^7 bit) per prevenire attacchi al sistema e, dato che dobbiamo inserire 10^4 testi cifrati nella chiave pubblica, quest'ultima risulterebbe lunga 10^{11} bit, circa 12.5 GB. Grazie invece alla riduzione proposta in [9], la dimensione della chiave pubblica diventa $2 \cdot 10^7$ bit, che corrispondono a 2.5 MB, il quale è certamente un valore molto più maneggevole, sebbene sia ancora piuttosto "grande".

Bibliografia

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

[2] M. Ajtai, *Generating hard instances of lattice problems. (Extended Abstract)*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 1996, 99-108, ACM.

[3] T. M. Apostol, *Introduction to analytic number theory*, Springer, 1976.

[4] T. M. Apostol, *A proof that Euler missed: evaluating* $\zeta(2)$ *the easy way*, Math. Intelligencer, vol. 5, no. 3, 1983, 59-60.

[5] Z. Brakerski, C. Gentry, V. Vaikuntanathan, *(Leveled) Fully Homomorphic Encryption without Bootstrapping*, ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, 309-325.

[6] Y. Chen, P. Q. Nguyen, *Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers*, Advances in Cryptology - EUROCRYPT 2012, Lecture Notes in Computer Science 7237, Springer, 2012, 502-519.

[7] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer, 1993.

[8] J.S. Coron, A. Mandal, D. Naccache, M. Tibouchi, *Fully Homomorphic Encryption over the Integers with Shorter Public Keys*, Advances in Cryptology CRYPTO 2011 - 31st Annual Cryptology Conference, Lecture Notes in Computer Science 6841, Springer, 2011, 487-504.

[9] J.S. Coron, D. Naccache, M. Tibouchi, *Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers*, Advances in Cryptology EUROCRYPT 2012, Lecture Notes in Computer Science 7237, Springer, 2012, 446-464.

[10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, *Fully homomorphic encryption over the integers*, Lecture Note in Computer Science 6110, 2010, 24-53, Springer.

[11] S. Galbraith, *Mathematics of public key cryptography*, Cambridge U.P, 2012

[12] C. Gentry, *A fully homomorphic encryption scheme*, PhD thesis, Stanford University, 2009.

[13] C. Gentry, *Computing arbitrary functions of encrypted data*, Comm. of ACM **53**, 2010, 97-105.

[14] C. Gentry, *Key Recovery and Message Attacks on NTRU-Composite*, Lecture Notes in Computer Science 2045 (2001), 182-194, Springer.

[15] C. Gentry, S. Halevi, *Implementing Gentry's Fully-Homomorphic Encryption Scheme*, Cryptology ePrint Archive, 2010.
C. Gentry, S. Halevi, *Implementing Gentry's Fully-Homomorphic Encryption Scheme*, Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology, EUROCRYPT'11, 2011, 129-148

[16] C. Gentry, S. Halevi, *Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits*, Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS '11, 2011, 107-109.

[17] S. Goldwasser, D. Micciancio, *Complexity of Lattice Problems*, Kluwer, 2002.

[18] B. Hayes, *Alice and Bob in cipherspace*, American Scientist **100**, 2012, 362-367.

[19] J. Hoffstein, J. Pipher, J. H. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem*, Lecture Notes in Computer Science 1423, 1998, 267-288, Springer.

[20] N. Howgrave-Graham, *Approximate integer common divisors*, CaLC 2001, LNCS, 2001, 51-66, Springer.

[21] T. Laarhoven, J. van de Pol, B. de Weger, *Solving Hard Lattice Problems and the Security of Lattice-Based Cryptosystems*, IACR Cryptology ePrint Archive, 2012.

[22] K. Lauter, M. Naehrig, V. Vaikuntanathan, *Can Homomorphic Encryption be practical?*, Proceedings of the 3rd ACM workshop on Cloud computing security workshop, CCSW '11, 2011, 113-124.

[23] Y. K. Liu, V. Lyubashevsky, D. Micciancio, *On Bounded Distance Decoding for general lattice*, Lecture Notes in Computer Science 4110, 2006, 450-461, Springer.

[24] V. Lyubashevsky, D. Micciancio, *On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem*, Lecture Notes in Computer Science 5677, 2009, 577-594, Springer.

[25] M. S. Paterson, L. J. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials*, SIAM J. Comput. 2, 1973, 60-66.

[26] R. L. Rivest, L. Adleman, M. L. Dertouzos, *On data banks and privacy homomorphism*, in "Foundations of Secure Computation", Academic Press, 1978, 169-180.

[27] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, STOC '05, 2005, 84-93.

[28] V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge U. P., 2005.

[29] N.P. Smart, F. Vercauteren, *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*, Cryptology ePrint Archive, 2009.
N.P. Smart, F. Vercauteren, *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*, Public Key Cryptography - PKC 2010, Lecture Notes in Computer Science Volume 6056, Springer, 2010, 420-443.

[30] J. Talbot, D. Welsh, *Complexity and Cryptography. An Introduction*, Cambridge University Press, 2006.