UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

GGH: un criptosistema basat en reticles

Autor: Johana Chen

Director: Dr. Xavier Guitart Morales
Realitzat a: Facultat de Matemàtiques i Informàtica

Barcelona, 13 de juny de 2023

Abstract

In this project we will study and implement the GGH public key cryptosystem, a lattice-based cryptosystem. We will introduce the basic definitions and properties of the lattice theory and the lattice-based problems on which the GGH cryptosystem is based on. Then, we will present two algorithms with lattices, the first one essential for the development of the cryptosystem and the second one useful for attacks against its security. Finally, we will describe in detail the GGH cryptosystem and perform the practical implementation.

Resum

En aquest treball estudiarem i posarem en pràctica el criptosistema de clau pública GGH, un criptosistema basat en reticles. Introduïrem les definicions i propietats bàsiques de la teoria de reticles i els problemes basats en reticles en els quals es fonamenta el criptosistema GGH. A continuació, presentarem dos algorismes amb reticles, la primera essencial per el desenvolupament del criptosistema i la segona útil per els atacs contra la seva seguretat. Finalment, descriurem amb detall el criptosistema GGH i realitzarem la implementació pràctica.

2020 Mathematics Subject Classification. 11H06, 94A60, 15A03, 46A40

Agraïments

En primer lloc, vull agrair al meu tutor, Dr. Xavier Guitart, per la seva dedicació, la paciència i tota l'ajuda al llarg del treball, per introduir-me un tema tan interessant i voler adaptar el treball a un àrea que m'és més còmode.

També, agraeïxo de tot cor a la meva família, per tot els ànims i suport que m'han donat durant tota la vida i en especial durant la carrera. Per estar sempre disponibles per mi.

Finalment, vull donar les gràcies a tots el professors, amics i companys per fer d'aquests anys una expreriència innolvidable. En especial als meus amics, que sempre han estat disposats a ajudar, no només en l'estudi sinó també per donar suport emocional.

Índex

Capítol 1

Introducció

1.1 Introducció

Actualment vivim en la era digital o també coneguda com la era de la informació, una era que gràcies al desenvolupament de les noves tecnologies, la comunicació i les TICs, la informació corre encara més ràpid que el moviment físic. És per això que garantir la seguretat de la informació ha pres un paper clau en el món de la tecnologia. I és aquí on entra en joc la criptografia, l'art d'escriure amb clau secreta o d'una manera enigmàtica.

Pot semblar que la criptografia va néixer amb l'evolució de la tecnologia, però en realitat el seu origen va ser milers d'anys enrere. Podríem situar l'origen de la criptografia a l'Antic Egipte, fa més de 4500 anys, on alguns egipcis feien ús dels jeroglífics per comunicar-se entre ells. Però no es pot afirmar que aquests fossin intents de comunicació secreta, més aviat es considera que l'objectiu era donar un toc de misteri i diversió al lector. Així, estríctament parlant, podríem dir que la criptografia es va generar a l'Antiga Grècia, al segle 5 a.C, de quan queden registrats els primers missatges encriptats. Els espartans feien ús del sistema de codificació conegut com l'escílata, un sistema format per dues vares de gruix similar i una tira de cuir o papir. L'emisor només havia d'enrotllar la cinta a un dels bastons i escriure el missatge longitudinalment de manera que a cada volta de la cinta aparegués una lletra del missatge. Així, jugant amb el diàmetre de la vara es podia encriptar i desencriptar missatges.

Amb el pas dels anys, la criptografia també ha anat evolucionant i adaptant diferents sistemes de xifratge. Va ser al voltant de l'any 1000 quan va aparèixer la tècnica de l'anàlisi de freqüències que va trencar els xifrats per substitució monoalfabètics. Aquest fet va comportar l'avenç criptoanalític més important fins a la Segona Guerra Mundial i va donar inici a la criptografia medieval.

Alguns dels fets més destacats en la història de la criptografia van ser el xifratge Cèsar, una de les tècniques de xifratge clàssic més senzilles però més conegudes; l'ús de la criptografia a la Primera Guerra Mundial a través del telegrama Zimmermann que va accelerar l'entrada del Estats Units a la guerra; i els missatges xifrats de l'Alemanya nazi durant la Segona Guerra Mundial, que podrien haver escurçat la guerra fins a dos anys si haguèssin sigut capaços de desxifrar-los.

Finalment, la criptografia moderna va ser introduïda l'any 1948, amb la Teoria de la Informació per Claude Shannon, considerat el pare de la criptografia matemàtica. Fins aleshores, tots els criptosistemes eren de clau privada, és a dir, l'emisor i el receptor

havien de compartir una mateixa clau, desconeguda per qualsevol altra persona, per poder xifrar i desxifrar els missatges. Això suponia que d'alguna manera l'emisor havia de fer arribar la clau privada al receptor. L'arribada de la criptografia moderna va fer possible desenvolupar criptosistemes de clau pública. El mètode de clau pública, o també coneguda com asimètrica, va ser idealitzat per Whitfield Diffie i Martin Hellman. A diferència dels criptosistemes de clau privada, els de clau pública proposen l'ús de dues claus relacionades matemàticament: una clau pública i una clau privada. L'emisor ha de xifrar el missatge a enviar amb la clau pública mentre que el receptor ha de desxifrar el missatge amb la seva clau privada. Encara que es coneix el text xifrat i la clau pública, aquesta clau no és capaç de desencriptar el text, per tant, sempre que la clau privada quedi protegida i secreta, el missatge a transmitir queda protegit. A més, el receptor pot canviar la seva clau privada i publicar la nova clau pública en qualsevol moment. Els criptosistemes de clau pública més coneguts són el RSA, un criptosistema basat en la factorització de nombres enters; el DSA, una variant més eficient del ElGamal que també s'utilitza per firmar; i Diffie-Hellman, un protocol d'intercanvi de claus.

L'any 1994, Peter Shor va proposar l'algorisme de Shor, un algorisme quàntic que realitza la descomposició en factors d'un nombre N en un temps $\mathcal{O}((\log N)^3)$. L'algorisme utilitza tècniques característiques de la computació quàntica per a poder reduir la descomposició de factors d'un ordre de complexitat exponencial a un de polinomial. Així, amb aquest algorisme i l'aparició dels ordinadors quàntics la seguretat dels criptosistemes moderns de fins aleshores van quedar greument amenaçades. És més, l'any 2016, Issac Chuang, juntament amb un grup d'investigadors del MIT, van aconseguir crear un ordinador quàntic de 5 qubits capaç de córrer l'algorisme de Shor per factoritzar el nombre 15 amb una certesa del 99%. Tot i que de moment encara no s'ha arribat a construir ordinadors quàntics suficientment eficients per trencar aquests criptosistemes, hi ha especialistes que afirmen que amb els avenços tecnològics és només qüestió de temps que s'arribi a trencar definitivament la gran majoria dels criptosistemes de clau pública que s'utilitza en l'actualitat. Encara que no hi hagi certesa d'aquest fet, les entitats tecnològiques han donat molta importància a aquesta possible amenaça. En resposta a aquest atac, es va desenvolupar la criptografia postquàntica, és a dir, criptosistemes resistents als atacs dels ordinadors quàntics.

Aquest fet va captar gran atenció dels acadèmics i la indústria i més recentment de diversos Tallers del *European Telecommunications Standards Institute* (ETSI). En concret, l'any 2016, l'Institut Nacional d'Estàndards i Tecnologia (NIST de l'anglès *National Institute of Standards and Technology*) va iniciar un procès d'estandardització d'algorismes criptogràfics de clau pública amb resistència quàntica. Actualment, el concurs ja ha arribat a la quarta i última ronda de submissions i podem trobar tres criptosistemes finalistes, BIKE, Classic McEliece i HQC.

Dins de la criptografia postquàntica trobem famílies d'algorismes que destaquen per ser les més prometedores en garantir la seguretat postquàntica. Aquestes són, la criptografia basada en codi (CBC de l'anglès *Code-Based Cryptograaophy*), la criptografia basada en hash (HBC de l'anglès *Hash-Based Cryptography*), la criptografia basada en funcions polinomials multivariables (MVC de l'anglès *Multivariate-Based Cryptography*), el xifrat simètric basat en la clau secreta de Rijndael i la criptografia basada en reticles (LBC de l'anglès *Lattice-Based Cryptography*). Totes aquestes tècniques són quànticament segurs, per tant, no s'ha pogut aplicar l'algorisme de Shor en cap dels algorismes anteriors.

De les tècniques mencionades hi destaca la criptografia basada en reticles. Els reticles són espais n-dimensional generats per combinacions lineals enteres de vectors d'una base

amb coeficients en el enters. Aquests criptosistemes destaquen per gaudir d'una bona seguretat, per tenir implementacions relativament eficients i per la seva simplicitat. El criptosistema GGH és un dels criptosistemes basats en reticles que més renom té, ja que es considera un dels criptosistemes que va introduir la criptografia basada en reticles. El seu origen va ser anterior a l'algorisme de Shor, quan encara no existia el concepte de criptografia postquàntica. És per això que en aquella època el criptosistema no va triunfar ja que no superava l'eficiència dels altres criptosistemes, com per exemple el RSA. Actualment, amb la introducció de la criptografia postquàntica, el criptosistema GGH ha guanyat molta més importància per ser una bona candidata com a criptosistema resistent als atacs quàntics. És exactament aquest el motiu pel qual volem dedicar aquest treball a l'estudi del criptosistema GGH i les possibles amenaces que existeixen.

1.2 Estructura de la Memòria

El principal objectiu del treball és l'estudi del criptosistema GGH, englobant la teoria de reticles, els algorismes amb reticles i la criptografia basada en reticles. La memòria del treball consta de quatre capítols.

Al capítol 1 introduirem la definició formal de reticle seguida d'una sèrie de definicions i propietats necessàries per el desenvolupament del treball. A més, presentarem els problemes de vectors en reticles, coneguts per ser problemes complicats de resoldre que garanteixen la seguretat dels criptosistemes basats en reticles.

A continuació, al capítol 2 parlarem dels algorismes amb reticles. En aquest capítol presentem dos tipus d'algorismes, per un costat l'algorisme de Babai essencial per el procés de desxifratge del criptosistema GGH, i per l'altre costat els algorismes de reducció de reticles, algorismes que es consideren amenaces contra la seguretat del criptosistema. D'aquest segon tipus d'algorismes cal destacar l'algorisme LLL per ser l'algorisme que ha donat l'atac més important contra la seguretat dels criptosistemes basats en reticles. L'algorisme va ser creat per Arjen Lenstra, Hendrik Lenstra i László Lovász, l'any 1982. Destaquem especialment a László Lovász, matemàtic hongarès conegut per la seva contribució en la branca de la combinatòria, qui va rebre el premi Abel l'any 2021.

Seguidament, al capítol 3 presentem el tema principal del treball, el criptosistema GGH. Farem una breu introducció sobre la criptografia basada en reticles i passarem a estudiar en detall el criptosistema GGH. A més, realitzarem una aplicació de l'algorisme LLL, l'algorisme de reducció de reticles estudiat al capítol anterior, sobre el criptosistema GGH.

Finalment, al capítol 5 explicarem la part pràctica del treball, que consta de la programació en Sage del criptosistema GGH i de l'algorisme LLL.

Capítol 2

Reticles

2.1 Preàmbuls

Abans d'introduir els reticles, farem un breu recordatori sobre algunes definicions i propietats importants dels espais vectorials.

En aquest capítol considerarem espais vectorials continguts en $\mathbb{R}^m$ per $m \in \mathbb{Z}^+$. Recordem que un subespai vectorial V és un subconjunt de $\mathbb{R}^m$ tal que

$$\alpha_1 v_1 + \alpha_2 v_2 \in V \quad \text{per a tot } v_1, v_2 \in V \text{ i tot } \alpha_1, \alpha_2 \in \mathbb{R}.$$

És a dir, un subespai vectorial és un subconjunt de $\mathbb{R}^m$ tancat per la suma i la multiplicació per escalars de $\mathbb{R}$.

A l'hora d'introduir els reticles, cal tenir present els conceptes de combinació lineal, independència lineal i les propietats principals d'una base d'un conjunt V.

Quan treballem amb reticles, trobar una bona base és molt important. El criteri amb el qual determinem si una base és bona o no és segons la seva ortogonalitat. L'algorisme de Gram-Schimdt és un mètode estàndard per crear una base ortonormal. Pel nostre propòsit és suficient que l'algorisme trobi una base ortogonal.

Teorema 2.1.1. ***(Algorisme de Gram-Schmidt)*** *Sigui $\{v_1, v_2, \ldots, v_n\}$ una base de l'espai vectorial $V \subset \mathbb{R}^m$. L'algorisme crea una base ortogonal $\{v_1^*, v_2^*, \ldots, v_n^*\}$ de V.*

Algorithm 1 Algorisme de Gram-Schmidt

$v_1^* = v_1$
for $i = 2, 3, \ldots, n$ **do**
  $\mu_{ij} = v_i \cdot v_j^* / \|v_j^*\|^2 \quad \text{per } 1 \leq j < i$
  $v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{ij} v_j^*$
end for

2.2 Reticles. Definicions bàsiques i propietats

Un cop fet el recordatori sobre espais vectorials, presentem la definició formal de reticle i les seves propietats.

Definició 2.2.1. *Sigui $v_1, \ldots, v_n \in \mathbb{R}^m$ un conjunt de vectors linealment independents. El reticle L generat per $\{v_1, \ldots, v_n\}$ és el conjunt de combinacions lineals dels vectors $v_1, \ldots, v_n$ amb coeficients en $\mathbb{Z}$*

$$L = \{a_1 v_1 + a_2 v_2 + \cdots + a_n v_n : a_1, a_2, \ldots, a_n \in \mathbb{Z}\}.$$

Una base de L és qualsevol conjunt de vectors independents que generi L. Dues bases de L tindran el mateix nombre d'elements. La dimensió de L és el cardinal de la base de L.

Suposem que $\{v_1, \ldots, v_n\}$ és una base del reticle L i $w_1, \ldots, w_n \in L$ un conjunt de vectors de L. Sabem que w_j es pot expressar com una combinació lineal dels vectors de la base,

$$\begin{aligned} w_1 &= a_{11} v_1 + a_{12} v_2 + \cdots + a_{1n} v_n, \\ w_2 &= a_{21} v_1 + a_{22} v_2 + \cdots + a_{2n} v_n, \\ &\vdots \\ w_n &= a_{n1} v_1 + a_{n2} v_2 + \cdots + a_{nn} v_n. \end{aligned}$$

Ara, com que són vectors del reticle L, sabem que els coeficients a_{ij} són enters.
Suposem que volem expressar els vectors v_i en funció de w_j. Això implica trobar la matriu inversa de

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}. \tag{2.2.1}$$

Com que necessitem que els v_i estiguin expressats com a combinació lineal dels w_j, A^{-1} haurà de tenir entrades enteres. En conseqüència,

$$1 = \det(I) = \det(AA^{-1}) = \det(A)\det(A^{-1})$$

on $\det(A)$ i $\det(A^{-1})$ són enters, per tant, $\det(A) = \pm 1$. De la mateixa manera, si $\det(A) = \pm 1$, pel teorema de la matriu adjunta, A^{-1} tindrà entrades enteres. Això prova el següent resultat.

Proposició 2.2.2. *Dues bases d'un reticle L estan relacionades per una matriu amb coeficients enters i determinant igual a ± 1.*

Definició 2.2.3. *Diem que el subconjunt $L \subset \mathbb{R}^m$ és un subgrup additiu si és tancat per la suma i la resta. Anomenem subgrup additiu discret si existeix una constant positiva $\epsilon > 0$ que s'atisfà: per a tot $v \in L$,*

$$L \cap \{w \in \mathbb{R}^m : \|v - w\| < \epsilon\} = \{v\}.$$

És a dir, prenent qualsevol vector $v \in L$ i prenent una bola de radi ϵ i centre v, la bola no conté cap element de L diferent de v.

Definició 2.2.4. *Un reticle enter és un reticle tal que tots els seus vectors tenen coordenades enteres. Equivalentment, un reticle enter és un subgrup additiu de $\mathbb{Z}^m$ per algun $m \geq 1$.*

Amb la definició de subgrup additiu discret, podem donar una definició alternativa de reticle, una definició més abstracta, que combina l'àlgebra i la geometria.

Teorema 2.2.5. *Un subconjunt de $\mathbb{R}^m$ és un reticle si i només si és un subgrup additiu discret.*

Demostració. Aquest resultat no és essencial per el desenvolupament del treball, deixem la demostració per el lector. També es pot consultar la demostració a la Proposició 4.2 de [2]. □

Definició 2.2.6. *Sigui L un reticle de dimensió n i $\{v_1, v_2, \ldots, v_n\}$ una base de L. El domini fonamental o paral·lelepípede fonamental de L corresponent a aquesta base és el conjunt*

$$\mathcal{F}(v_1, \ldots, v_n) = \{t_1 v_1 + t_2 v_2 + \cdots + t_n v_n : 0 \leq t_i < 1\}.$$

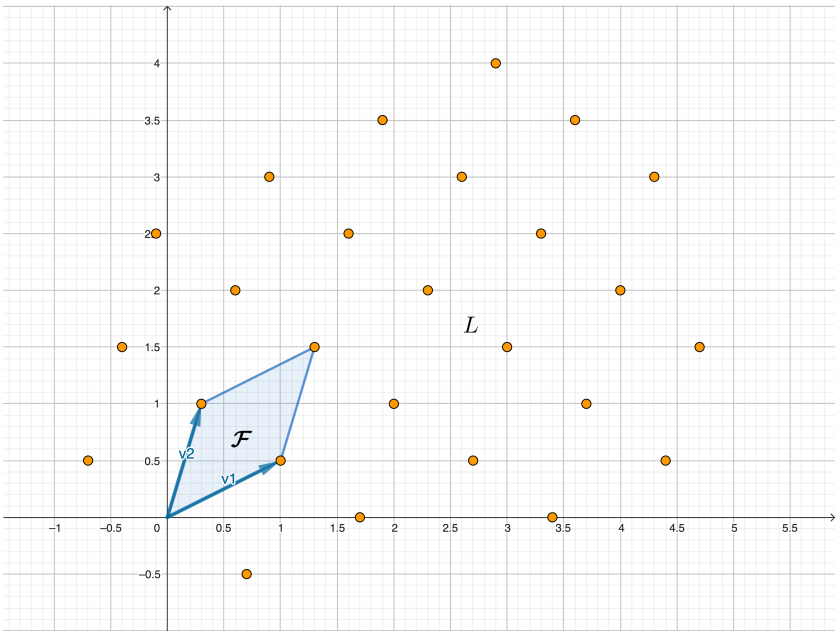

Figura 2.1: Un domini fonamental $\mathcal{F}$ del reticle L

Proposició 2.2.7. *Sigui $L \subset \mathbb{R}^n$ un reticle de dimensió n i $\mathcal{F}$ el domini fonamental de L. Qualsevol vector $w \in \mathbb{R}^n$ es pot escriure de la forma*

$$w = t + v$$

per uns únics $t \in \mathcal{F}$ i $v \in L$.
Equivalentment, la unió dels dominis fonamentals traslladats

$$\mathcal{F} + v = \{t + v : t \in \mathcal{F}\},$$

on v recorre tots els elements de L, recobreix $\mathbb{R}^n$. Veieu la Figura 2.2

Demostració. Primer demostrem que efectivament w es pot escriure de la forma $w = t+v$. Sigui $\{v_1, \ldots, v_n\}$ una base del reticle L que dona el domini fonamental $\mathcal{F}$. Els vectors de la base $\{v_1, \ldots, v_n\}$ són linealment independents en $\mathbb{R}^n$, per tant també és base en $\mathbb{R}^n$. Això vol dir que podem expressar qualsevol $w \in \mathbb{R}^n$ de la forma

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n,$$

per algun $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$.

Separem la part decimal de cada α_i tal que

$$\alpha_i = t_i + a_i$$

amb $0 \leq t_i < 1$ i $a_i \in \mathbb{Z}$.

Per tant, efectivament, podem escriure w com

$$w = t_1 v_1 + t_2 v_2 + \cdots + t_n v_n + a_1 v_1 + a_2 v_2 + \cdots + a_n v_n \in \mathcal{F} + L.$$

Ara provem la seva unicitat. Suposem que $w = t + v = t' + v'$, on $t, t' \in \mathcal{F}$ i $v, v' \in L$. Llavors,

$$(t_1 + a_1)v_1 + (t_2 + a_2)v_2 + \cdots + (t_n + a_n)v_n = (t'_1 + a'_1)v_1 + (t'_2 + a'_2)v_2 + \cdots + (t'_n + a'_n)v_n$$

Com que $v_1, \ldots, v_n$ són linealment independents, tenim que

$$t_i + a_i = t'_i + a'_i \quad \text{per a tot } i = 1, 2, \ldots, n.$$

Per tant,

$$t_i - t'_i = a'_i - a_i \in \mathbb{Z}$$

és un enter. A més sabem que t_i i t'_i satisfan $0 \leq t_i < 1$ i $0 \leq t'_i < 1$, per tant $t_i - t'_i$ serà enter únicament quan $t_i = t'_i$. És a dir, $t = t'$ i conseqüentment

$$v = w - t = w - t' = v'.$$

Això demostra la unicitat de $t \in \mathcal{F}$ i $v \in L$. □

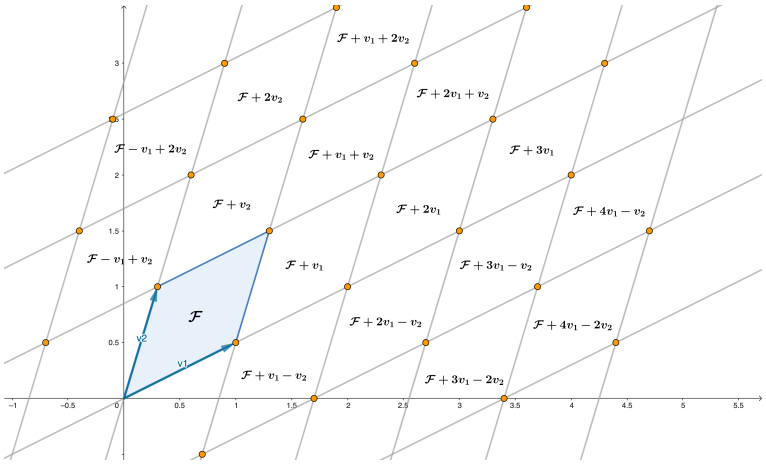

Figura 2.2: Dues bases diferents per un mateix reticle L

A continuació introduïm el volum d'un domini fonamental del reticle L que es tracta d'una invariant del reticle molt important.

Definició 2.2.8. *Sigui L un reticle de dimensió n i $\mathcal{F}$ un domini fonamental de L. El volum n-dimensional de $\mathcal{F}$ s'anomena el determinant de L, també es coneix com el covolum* [1] *de L. El covolum de L es denomina* $\det(L)$.

Donada la base $\{v_1, \ldots, v_n\}$, si la imaginem com els vectors que determinen els costats del paral·lepípede $\mathcal{F}$, llavors el volum màxim es donarà quan els vectors són ortogonals dos a dos. Aquest raonament ens porta a la següent desigualtat.

Proposició 2.2.9. ***(Desigualtat de Hadamard)*** *Sigui L un reticle, sigui $\{v_1, \ldots, v_n\}$ qualsevol base de L i $\mathcal{F}$ un domini fonamental de L. Llavors*

$$\det L = \mathrm{Vol}(\mathcal{F}) \leq \|v_1\|\|v_2\| \cdots \|v_n\|.$$

Com més ortogonal sigui la base, més s'aproparà la desigualtat de Hadamard a ser una igualtat.

Abans de desenvolupar la demostració necessitem introduir primer la següent proposició, on s'especifica com es calcula el volum del domini fonamental $\mathrm{Vol}(\mathcal{F})$.

Proposició 2.2.10. *Sigui $L \in \mathbb{R}^n$ un reticle de dimensió n, sigui $\{v_1, v_2, \ldots, v_n\}$ una base de L i $\mathcal{F} = \mathcal{F}(v_1, \ldots, v_n)$ el domini fonamental associat a aquesta base.*
Sigui $v_i = (r_{i1}, r_{i2}, \ldots, r_{in})$ les coordenades del vector i-èssim de la base i F la matriu formada per els vectors de la base en files

$$F = F(v_1, \ldots, v_n) = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{pmatrix}, \tag{2.2.2}$$

el volum de $\mathcal{F}$ és

$$\mathrm{Vol}(\mathcal{F}(v_1, \ldots, v_n)) = |\det(F(v_1, \ldots, v_n))|.$$

Demostració. Podem calcular el volum $\mathcal{F}$ de la següent manera:

$$\mathrm{Vol}(\mathcal{F}) = \int_{\mathcal{F}} dx_1 dx_2 \cdots dx_n.$$

Per definició del domini fonamental, podem fer un canvi de variables de $x = (x_1, \ldots, x_n)$ a $t = (t_1, \ldots, t_n)$ tal que

$$(x_1, x_2, \ldots, x_n) = t_1 v_1 + t_2 v_2 + \ldots + t_n v_n.$$

En termes de la matriu F (2.2.2), el canvi de variable ve representada per l'equació $x = tF$, on la matriu jacobiana del canvi de variable és la matriu F. A més, com el domini fonamental $\mathcal{F}$ és la imatge de F en la unitat cúbica $C_n = [0,1]^n$, és a dir,

$$F : [0,1]^n \longrightarrow \mathcal{F} \subset \mathbb{R}^n,$$

[1] El reticle L, al ser una col·lecció de punts comptable, no té volum. Si $L \subset \mathbb{R}^n$ té dimensió n , llavors el covolum de L per definició és el volum del grup quocient $\mathbb{R}^n/L$.

$$(t_1, t_2, \ldots, t_n) \longmapsto t_1 v_1 + t_2 v_2 + \ldots + t_n v_n$$

per tant, $\mathcal{F} = F([0,1]^n) = FC_n$.
La fórmula de canvi de variable per integrals és

$$\int_{\mathcal{F}} dx_1 dx_2 \cdots dx_n = \int_{FC_n} dx_1 dx_2 \cdots dx_n = \int_{C_n} |\det F| dt_1 dt_2 \cdots dt_n$$
$$= |\det F| \mathrm{Vol}(C_n) = |\det F|.$$

□

Un cop demostrada aquesta proposició ja tenim les eines suficients per provar la desigualtat de Hadamard.

Demostració. Desenvoluparem la demostració fent ús de la descomposició QR. Sabem que qualsevol matriu A es pot descomposar de tal manera que

$$A = QR$$

on Q és una matriu ortogonal i R una matriu triangular superior. Per tant, en el nostre cas, la matriu que té per columnes els vectors de la base F es pot descomposar com

$$F = QR.$$

En particular, podem prendre els vectors de la base ortonormal que hem obtingut a través de Gram-Schmidt, $v_1^*, \ldots, v_n^*$, com a columnes de la matriu Q. Sigui r_{ij} els coeficients de la matriu R, tenim que

$$v_j = \sum_{i=1}^{j} r_{ij} v_i^*.$$

Prenent la norma al quadrat obtenim

$$\|v_j\|^2 = \sum_{i=1}^{j} |r_{ij}^2| \|v_j^*\|^2 = \sum_{i=1}^{j} |r_{ij}|^2 \geq |r_{jj}|^2 \longrightarrow |r_{jj}| \leq \|v_j\|,$$

on hem aplicat l'ortonormalitat dels vectors v_j^*. Finalment, tenim

$$\mathrm{Vol}(\mathcal{F}) = |\det(F)| = |\det(QR)| = |\det(Q)| \cdot |\det(R)| = 1 \cdot \left| \prod_{j=1}^{n} r_{jj} \right| = \prod_{j=1}^{n} |r_{jj}| \leq \prod_{j=1}^{n} \|v_j\|.$$

Tal i com volíem demostrar.

També es pot trobar una altra versió de la demostració al Teorema 19.13 de [7] □

Corol·lari 2.2.11. *Sigui $L \subset \mathbb{R}^n$ un reticle de dimensió n. Tot domini fonamental de L tindrà el mateix volum. Per tant, el determinant de L és un element invariant del reticle L, independentment del domini fonamental escollit.*

Demostració. Sigui $\{v_1, \ldots, v_n\}$ i $\{w_1, \ldots, w_n\}$ dues bases del reticle L, i sigui $F(v_1, \ldots, v_n)$ i $F(w_1, \ldots, w_n)$ les matrius associades. Per la proposició 2.2.2 sabem que

$$F(v_1, \ldots, v_n) = AF(w_1, \ldots, w_n) \tag{2.2.3}$$

on A és una matriu $n \times n$ amb coeficients enters i $\det(A) = \pm 1$. Aplicant la proposició 2.2.10 obtenim

$$\begin{aligned}
\mathrm{Vol}(\mathcal{F}(v_1, \ldots, v_n)) &= |\det(\mathcal{F}(v_1, \ldots, v_n))| && \text{per la Proposició 2.2.10} \\
&= |\det(AF(w_1, \ldots, w_n))| && \text{per 2.2.3} \\
&= |\det(A)||\det(F(w_1, \ldots, w_n))| && \text{ja que } \det(AB) = \det(A)\det(B) \\
&= |\det(F(w_1, \ldots, w_n))| && \text{ja que } \det(A) = \pm 1 \\
&= \mathrm{Vol}(\mathcal{F}(w_1, \ldots, w_n)) && \text{per la Proposició 2.2.10}
\end{aligned}$$

□

2.3 Problemes de vectors curts en reticles

Quan parlem de problemes computacionals relacionats amb els reticles podem destacar dos problemes fonamentals. Per un costat, trobar el vector no nul més curt del reticle, i per l'altre, trobar el vector més proper a un vector donat que no pertany al reticle.
Dedicarem aquest apartat per presentar aquests dos problemes des d'un punt de vista teòric.

2.3.1 Problemes del vector més curt i del vector més proper

Primer fem una breu descripció dels dos problemes.

El problema del vector més curt: (SVP, *The Shortest Vector Problem*) Trobar un vector no nul més curt en un reticle L. És a dir, trobar $v \in L$ un vector no nul tal que la seva norma Euclideana $\|v\|$ sigui mínima.

El problema del vector més proper: (CVP, *The Closest Vector Problem*) Donat un vector $w \in \mathbb{R}^m$ tal que w no pertany a L, trobar un vector $v \in L$ més proper a w. És a dir, trobar $v \in L$ tal que la norma Euclideana $\|w - v\|$ sigui mínima.

Observació 2.3.1. Diem que SVP troba "un" vector amb norma Euclidiana mínima i no "el" vector ja que pot existir més d'un vector no nul del reticle que compleixi la condició. Per exemple, en $\mathbb{Z}^2$, els vectors $(0, \pm 1)$ i $(\pm 1, 0)$ són tots solucions de SVP. El mateix passa amb CVP.

A simple vista aquests problemes poden semblar molt senzills i inofensius, però en realitat són una gran amenaça per els criptosistemes basats en reticles. Veurem en els propers capítols que les solucions dels problemes SVP i CVP són eines molt útils per l'atac contra diversos criptosistemes.

En general, CVP és conegut per ser $\mathcal{NP}$-complet (o $\mathcal{NP}$-hard[2]) i SVP és també $\mathcal{NP}$-complet sota certes hipòtesis de reduccions aleatòries. Per entrar amb més detalls sobre la complexitat d'aquests problemes i les seves demostracions es pot consultar [6]. El que sí que cal esmentar és que, a mesura que augmenta la dimensió del reticle, els problemes

[2]La classe de complexitat $\mathcal{NP}$-hard es defineix com el conjunt dels problemes de decisió tals que si H és un problema d'aquesta classe, tot problema de $\mathcal{NP}$ es pot transformar en H en temps polinòmic.

SVP i CVP es fan cada cop més difícils de resoldre a nivell computacional. Encara així, tenen grans aplicacions en diferents àrees de la matemàtica pura i aplicada, fins i tot les solucions aproximades dels problemes en són molt útils.

A continuació descriurem una de les variants dels problemes SVP i CVP que seran crucials en el desenvolupament del treball.

Problema aproximat del vector més curt (apprSVP): Sigui L un reticle de dimensió n i sigui $\psi(n)$ una funció de n. Si $v_{shortest}$ és un dels vectors no nuls més curts del reticle L, trobar un vector no nul $v \in L$ tal que

$$\|v\| \leq \psi(n)\|v_{shortest}\|.$$

Observem que per a cada funció $\psi(n)$ tenim un diferent problema apprSVP. Depenent de la dimensió del reticle L hauríem de triar una funció $\psi(n)$ convenient.

Problema aproximat del vector més proper (apprCVP): Sigui L un reticle de dimensió n i sigui $\psi(n)$ una funció de n. Trobar un vector no nul $v \in L$ tal que

$$\|v\| \leq \psi(n).$$

2.3.2 Teorema d'Hermite i teorema de Minkowski

El SVP troba un dels vectors més curts del reticle L. Però com de curt és aquest vector? La longitud del vector depen de la dimensió i el determinant de L. Els següent resultat dona una fita superior a la norma Euclidiana del vector solució en funció de la dimensió i el determinant de L.

Teorema 2.3.2. ***(Teorema d'Hermite)*** *Tot reticle L de dimensió n conté un vector no nul $v \in L$ tal que*

$$\|v\| \leq \sqrt{n}\det(L)^{1/n}.$$

Observació 2.3.3. Donada la dimensió n, anomenem *constant d'Hermite* γ_n al valor mínim tal que tot reticle L de dimensió n conté un vector no nul $v \in L$ satisfent

$$\|v\|^2 \leq \gamma_n \det(L)^{2/n}.$$

Per tant, tenim que $\gamma_n \leq n$.
Per $1 \leq n \leq 8$ i $n = 24$ sabem exactament el valor de γ_n:

$$\gamma_2^2 = \frac{4}{3}, \quad \gamma_3^3 = 2, \quad \gamma_4^4 = 4, \quad \gamma_5^5 = 8, \quad \gamma_6^6 = \frac{64}{3}, \quad \gamma_7^7 = 64, \quad \gamma_8^8 = 256, \quad \gamma_{24} = 4.$$

De cara a la criptografia, estudiarem γ_n especialment en els casos on n és gran. Per valors grans de n la constant d'Hermite satisfà

$$\frac{n}{2\pi e} \leq \gamma_n \leq \frac{n}{\pi e}.$$

Observació 2.3.4. Existeixen altres versions del teorema d'Hermite que treballa amb més d'un vector. Per exemple, un reticle L n-dimensional sempre conté una base $\{v_1, \ldots, v_n\}$ tal que

$$\|v_1\|\|v_2\| \cdots \|v_n\| \leq n^{n/2}(\det L).$$

Aquesta desigualtat completa la desigualtat de Hadamard. (Proposició 2.2.9) que diu

$$\det L \leq \|v_1\|\|v_2\| \cdots \|v_n\|.$$

Definim *quocient de Hadamard de la base* $\mathcal{B} = (v_1, \ldots, v_n)$ com

$$\mathcal{H}(\mathcal{B}) = \left(\frac{\det L}{\|v_1\|\|v_2\| \cdots \|v_n\|} \right)^{1/n}.$$

Llavors, $0 < \mathcal{H}(\mathcal{B}) \leq 1$, com més proper a 1 sigui el valor del quocient de Hadamard, més ortogonal serà la base $\mathcal{B}$.

Per a realitzar la demostració del teorema d'Hermite fem ús del teorema de Minkowski.

Teorema 2.3.5. ***(Teorema de Minkowski)*** *Sigui $L \subset \mathbb{R}^n$ un reticle de dimensió n i $S \subset \mathbb{R}^n$ un conjunt convex simètric tal que el seu volum satisfà*

$$\mathrm{Vol}(S) > 2^n \det(L). \tag{2.3.1}$$

Llavors, S conté un vector no nul del reticle.
Si a més S és tancat, llavors serà suficient amb $\mathrm{Vol}(S) \geq 2^n \det(L)$.

Demostració. Sigui $\mathcal{F}$ un domini fonamental del reticle L. Sabem per la Proposició 2.2.7 que tot vector $a \in S$ es pot expressar de manera única com

$$a = v_a + w_a \qquad \text{on } v_a \in L \text{ i } w_a \in \mathcal{F}.$$

Dilatem S pel factor $\frac{1}{2}$, és a dir, fem S la meitat de petit,

$$\frac{1}{2}S = \left\{ \frac{1}{2}a : a \in S \right\},$$

i considerem l'apliació

$$\frac{1}{2}S \longrightarrow \mathcal{F}, \qquad \frac{1}{2}a \longmapsto w_{\frac{1}{2}a} \tag{2.3.2}$$

Si reduïm S per un factor de 2 llavors els volum també es reduïrà en un factor de 2^n, per tant

$$\mathrm{Vol}\left(\frac{1}{2}S\right) = \frac{1}{2^n}(\mathrm{Vol}S) > \det(L) = \mathrm{Vol}(\mathcal{F})$$

La desigualtat anterior ve donada per la hipòtesi que $\mathrm{Vol}(S) > 2^n \det(L)$ (2.3.1).

Com que S és fitat, aquest estarà recobert per un nombre finit de traslacions del domini fonamental $\mathcal{F}$. Per tant, l'aplicació 2.3.2 també estarà donada per un nombre finit de traslacions, fet que causa la conservació del volum de l'aplicació. Com hem vist que el domini $\frac{1}{2}S$ té un volum estríctament més gran que el volum de $\mathcal{F}$, podem afirmar que existeixen punts diferents $\frac{1}{2}a_1$ i $\frac{1}{2}a_2$ que tenen la mateixa imatge en $\mathcal{F}$. Per tant, podem dir que existeixen punts diferents de S tal que

$$\frac{1}{2}a_1 = v_1 + w \quad \text{i} \quad \frac{1}{2}a_2 = v_2 + w \qquad \text{amb } v_1, v_2 \in L \text{ i } w \in \mathcal{F}.$$

Combinant les dues igualtats obtenim

$$\frac{1}{2}a_1 - \frac{1}{2}a_2 = v_1 - v_2 \in L.$$

Observem que

$$\frac{1}{2}a_1 - \frac{1}{2}a_2$$

és el punt mig del segment que uneix a_1 i $-a_2$, on $-a_2$ és un punt de S per la simetria de S. Aquest punt mig també està contingut en S ja que S és convex.
Per tant,

$$0 \neq v_1 - v_2 \in S \cap L$$

és un vector no nul del reticle contingut en S.

Ara suposem que S és tancat i que $\text{Vol}(S) \geq 2^n \det(L)$. Per cada $k \geq 1$, expandim S a una proporció de $1 + \frac{1}{k}$ i apliquem el resultat anterior per trobar un vector no nul

$$0 \neq v_k \in \left(1 + \frac{1}{k}\right) S \cap L.$$

Com tots els vectors del reticle, $v_1, v_2, \ldots$ estan fitats pel conjunt $2S$ i L és discret, podem afirmar que el conjunt de vectors és finit. Ara, com $\left(1 + \frac{1}{k} S\right)$, per a tot $k \geq 1$ és una seqüència infinita, existeix un vector no nul $v \in L$ que apareix infinites vegades en la seqüència i que pertany a la intersecció

$$\bigcap_{k=1}^{\infty} \left(1 + \frac{1}{k}\right) S.$$

Sabem que S és tancat, per tant $\bigcap_{k=1}^{\infty} \left(1 + \frac{1}{k}\right) S = S$. Així, $0 \neq v \in S \cap L$. □

Demostració. (del Teorema d'Hermite) La demostració és una aplicació del teorema de Minkowski. Sigui $L \subset \mathbb{R}^n$ un reticle i sigui S un cub en $\mathbb{R}^n$ centrat a l'origen i amb costats de llargària $2B$,

$$S = \{(x_1, \ldots, x_n) \in \mathbb{R}^n : -B \leq x_i \leq B \quad \text{per a tot } 1 \leq i \leq n\}.$$

El conjunt S és fitat, tancat i simètric, amb volum

$$\text{Vol}(S) = (2B)^n.$$

Fixant $B = \det(L)^{1/n}$, tenim que $\text{Vol}(S) = 2^n \det(L)$. Amb aquesta hipòtesi, podem aplicar el teorema de Mikowski per deduir que existeix un vector no nul $0 \neq v \in S \cap L$. Expressant les coordenades de v com $v = (v_1, \ldots, v_n)$ i per com hem definit S tenim

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} \leq \sqrt{nB^2} = \sqrt{n}B = \sqrt{n}\det(L)^{1/n}.$$

□

Amb això completem la demostració del teorema d'Hermite.

Capítol 3

Algorismes amb reticles

3.1 L'algorisme de Babai

Sigui $L \subset \mathbb{R}^n$ un reticle amb una base $v_1, v_2, \ldots, v_n$ ortogonal, és a dir

$$v_i \cdot v_j = 0 \quad \text{per a tot } i \neq j,$$

la resolució dels problemes SVP i CVP seran molt senzilles.
Per a resoldre el problema del vector més curt SVP, observem que la longitud de qualsevol vector de L ve donada per la fórmula

$$\|a_1 v_1 + a_2 v_2 + \cdots + a_n v_n\|^2 = a_1^2\|v_1\| + a_2^2\|v_2\| + \cdots a_n^2\|v_n\|.$$

Com que $a_1, \ldots, a_n \in \mathbb{Z}$, el vector més curt de L serà el vector més curt del conjunt $\{\pm v_1, \ldots, \pm v_n\}$.

Suposem ara que volem trobar el vector més proper al vector donat $w \in \mathbb{R}^n$ en L. Expressem primer w com

$$w = t_1 v_1 + t_2 v_2 + \ldots, t_n v_n \quad \text{amb } t_1, \ldots, t_n \in \mathbb{R}.$$

Llavors, prenent $v = a_1 v_1 + a_2 v_2 + \ldots + a_n v_n \in L$, tenim que

$$\|v - w\|^2 = \|(a_1 - t_1)v_1 + (a_2 - t_2)v_2 + \ldots + (a_n - t_n)v_n\|$$

$$= (a_1 - t_1)^2\|v_1\|^2 + (a_2 - t_2)^2\|v_2\|^2 + \ldots + (a_n - t_n)^2\|v_n\|^2$$

Com $a_i \in \mathbb{Z}$, per minimitzar la distància entre v i w haurem de prendre a_i com l'enter més proper a t_i.
El mateix reonament es pot aplicar a qualsevol base de L.

En general, si els vectors de la base són bastant ortogonals, resoldre el problema del vector més proper CVP serà senzill. En aquest treball les anomenarem "bases bones". En cas contrari, l'algorisme anterior no serà eficient. Conseqüentment, les anomenarem "bases dolentes". A continuació farem un raonament geomètric per aquest argument.

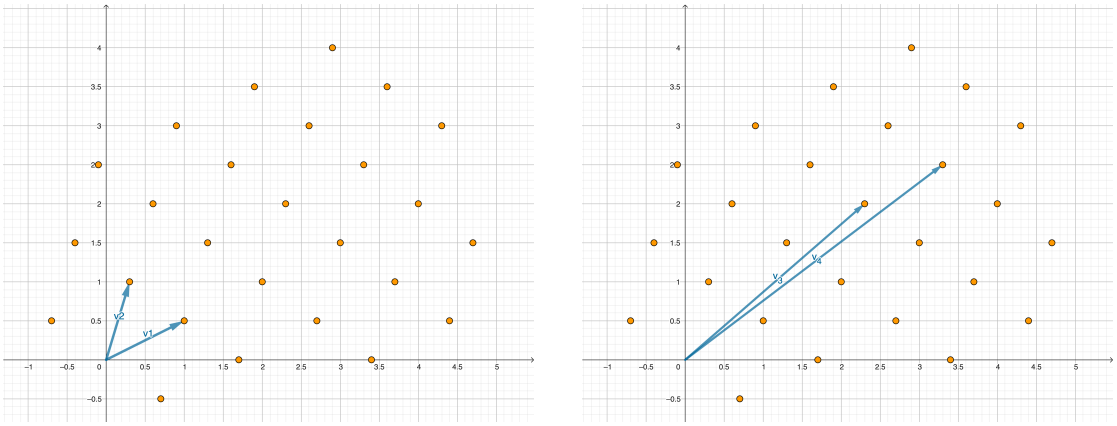

Figura 3.1: Exemple de "base bona" i "base dolenta" per el reticle L

Un domini fonamental $\mathcal{F}$ de L ve determinat per una base $\{v_1, \ldots, v_n\}$ de L. Segons la Proposició 2.2.10 sabem que $\mathcal{F}$ traslladat per elements de L cobreix tot l'espai $\mathbb{R}^n$. Per tant, qualsevol vector $w \in \mathbb{R}^n$ és contingut en una única traslació $\mathcal{F} + v$ per algun element $v \in L$. Prenem com a possible solució del problema del vector més proper CVP el vèrtex del paral·lelepípede $L + v$ més proper a w. Sabent que

$$w = v + \epsilon_1 v_1 + \epsilon_2 v_2 + \ldots + \epsilon_n v_n \quad \text{per algun } 0 \leq \epsilon_1, \epsilon_2, \ldots, \epsilon_n < 1,$$

per trobar el vector més proper a w simplement prenem $\epsilon_i = 0$ si aquest és més petit que $\frac{1}{2}$ i $\epsilon_i = 1$ si és més gran o igual que $\frac{1}{2}$.

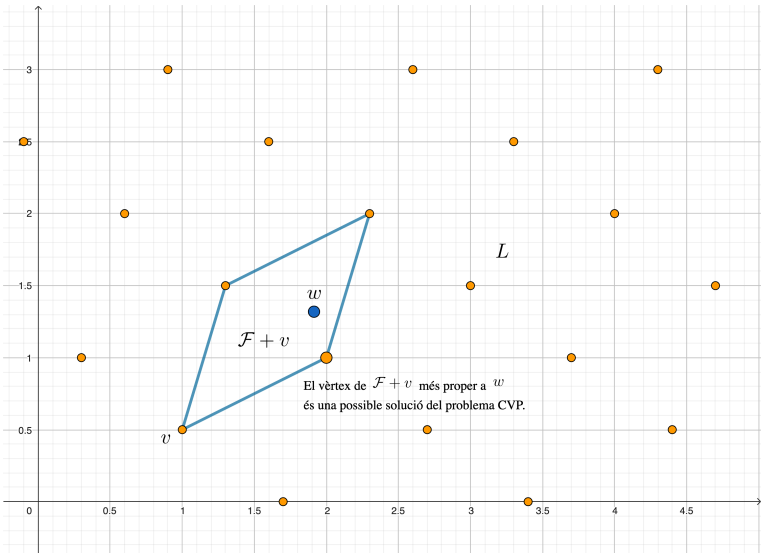

Figura 3.2: Solucionar CVP a partir d'un domini fonamental donat.

A la Figura 3.2 pot semblar que realitzant el procediment anterior podem resoldre el problema, però realment això és degut a que la base és bastant ortogonal.

Si intentem resoldre el problema del vector més proper CVP fent servir una base dolenta és molt probable que ens trobem amb problemes com per exemple el que es mostra a la Figura 3.3. Veiem que el punt objectiu fora del reticle és molt proper a un dels elements del reticle, en canvi, el vèrtex més proper del paral·lelepípede es troba ben lluny dels dos punts. Això és degut a que la base escollida no és gens ortogonal i l'angle entre els vectors de la base és molt petit. En dimensions superiors encara seria més greu el problema, l'algorisme anterior no seria capaç ni de resoldre apprCVP si la base no és ortogonal.

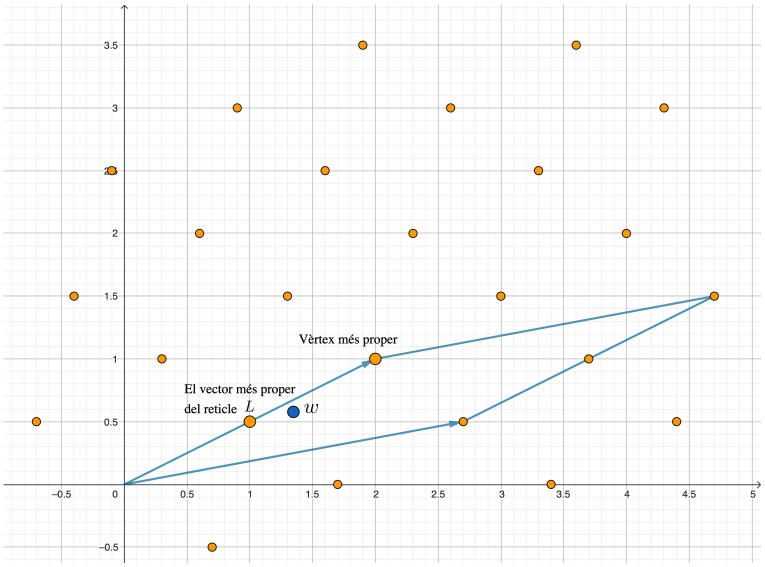

Figura 3.3: Algorisme de Babai sobre una "base dolenta"

Algorisme de Babai: Sigui $L \subset \mathbb{R}^n$ un reticle amb base $v_1, \ldots, v_n$, sigui $w \in \mathbb{R}^n$ un vector donat. Si els vectors de la base són suficientment ortogonals entre ells, llavors l'algorisme de Babai resol el problema del vector més proper CVP.

Algorithm 2 Algorisme de Babai

$w = t_1v_1 + t_2v_2 + \ldots + t_nv_n$ on $t_1, \ldots, t_n \in \mathbb{R}$
for $i = 1, 2, \ldots, n$ **do**
  $a_i = \lfloor t_i \rceil$
end for
return $v = a_1v_1 + a_2v_2 + \ldots + a_nv_n$

Sempre que els vectors de la base siguin raonablement ortogonals entre ells l'algorisme de Babai pot resoldre el problema aproximat del vector més proper, apprCVP. En cas que els vectors no siguin ortogonals l'algorisme retornarà un vector llunyà al vector del reticle més proper a w.

Observació 3.1.1. A l'hora de parlar de vectors suficientment ortogonals pot semblar molt ambigu el criteri en el que ens basem per determinar el seu grau d'ortogonalitat. A l'apartat (5.3.1) tractarem aquest tema estudiant els resultats experimentals de l'aplicació pràctica.

3.2 Algorismes de reducció de reticles

En aquest apartat ens dedicarem a estudiar l'algorisme LLL, un algorisme que resol el problema apprCVP en C^n operacions, on C és una constant petita i n és la dimensió del reticle L. Donem importància a l'algorisme LLL perquè és una possible amenaça per la seguretat dels criptosistemes basats en reticles. Encara que l'algorisme és capaç de resoldre SVP i CVP en dimensions petites, de moment encara no ho és en dimensions grans.

Abans d'introduir l'algorisme LLL, veurem l'algorisme de reducció de reticles de Gauss, que resol SVP en reticles de dimensió 2.

3.2.1 L'algorisme de reducció de reticles de Gauss

El propòsit principal de l'algorisme de Gauss és trobar una base del reticle L tal que la longitud dels seus vectors sigui la mínima. La idea general de l'algorisme és anar restant alternativament un múltiple del vector més curt de la base a l'altre vector fins que aquest no es pugui fer més petit.

Sigui $L \subset \mathbb{R}^2$ un reticle de dimensió 2 amb base $\{v_1, v_2\}$. Podem suposar que $\|v_1\| < \|v_2\|$, en cas contrari només caldria intercanviar v_1 i v_2. Volem fer més petit el vector v_2 restant-li un múltiple del vector v_1. Si li poguéssim restar qualsevol multiple de v_1, li retaríem $\frac{v_1 \cdot v_2}{\|v_1\|^2}$ vegades el vector v_1 i el nou vector quedaria com

$$v_2^* = v_2 - \frac{v_1 \cdot v_2}{\|v_1\|^2} v_1,$$

que és ortogonal a v_1. Alternativament, podem dir que v_2^* és la projecció de v_2 sobre la component ortogonal de v_1. (Figura 3.4)

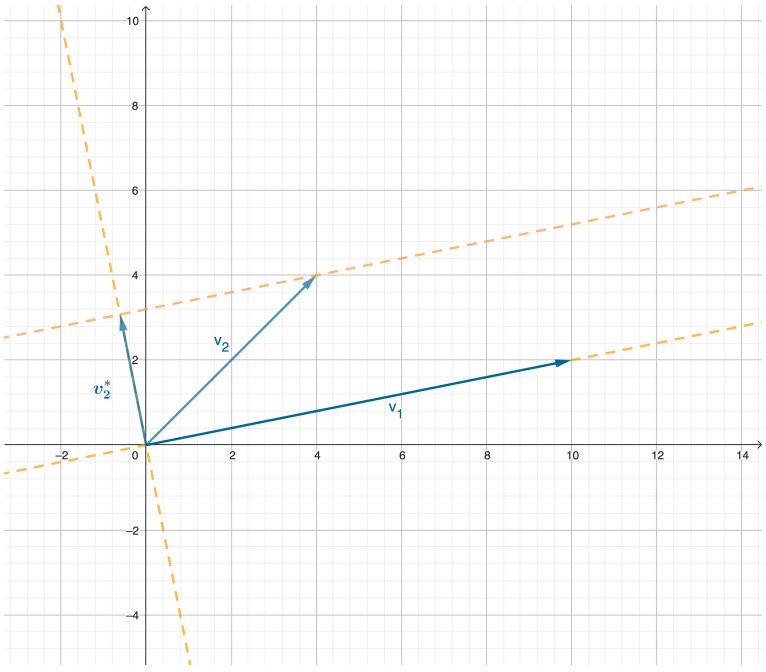

Figura 3.4: v_2^*, projecció de v_2 sobre la ortogonal complementària de v_1.

Fent aquest canvi, veiem que és molt probable que el nou vector v_2^* no pertanyi al reticle L, ja que pot no tenir coeficients enters. Per solventar aquest problema, imposem que el múltiple de v_1 sigui un enter. Per tant,

$$v_2^* = v_2 - mv_1 \quad \text{on } m = \left\lfloor \frac{v_1 \cdot v_2}{\|v_1\|^2} \right\rfloor.$$

Si un cop fet aquest canvi el nou vector v_2^* segueix sent més llarg que v_1, parem. En cas contrari intercanviem v_1 i v_2 i repetim el procés. Gauss va provar que l'algoritme acaba i troba una base molt bona de L.

Reducció Gaussiana de reticles: Sigui $L \subset \mathbb{R}^2$ un reticle de dimensió 2 amb base v_1, v_2. El següent algoritme acaba i troba una bona base per L.

Algorithm 3 Algorisme de Gauss

while $m \neq 0$ **do**
  if $\|v_2\| < \|v_1\|$ **then**
    $v^* = v_1$ ▷ Intercanviar v_1 i v_2
    $v_1 = v_2$
    $v_2 = v^*$
  end if
  $m = \left\lfloor \frac{v_1 \cdot v_2}{\|v_1\|^2} \right\rfloor$
  $v_2 = v_2 - mv_1$
end while
return v_1, v_2

L'algorisme acaba retornant el vector v_1 que és un dels vectors més curts del reticle L. Per tant, podem dir que l'algorisme de Gauss resol SVP. A més, l'algle θ entre els vectors v_1 i v_2 satisfà la desigualtat $|\cos\theta| \leq \frac{\|v_1\|}{2\|v_2\|}$.

Demostració. La demostració de l'algorisme de Gauss consta de dues parts. La primera part es tracta de demostrar que l'algorisme finalitza i retorna els dos vectors v_1 i v_2. Deixem aquesta part pel lector o també es pot consultar al Capítol 17 del llibre *Mathematics of public key cryptography* [3]. Ens dedicarem només a fer la demostració de la segona part. Veurem que el vector v_1 és efectivament un dels vectors no nuls més curts del reticle L, i que pertant l'algorisme resol SVP.

Suposem que l'algorisme finalitza i retorna els vectors v_1 i v_2. Sabem que al sortir de l'algorisme els vectors satisfan la desigualtat $\|v_2\| \geq \|v_1\|$. A més, com que l'algorisme ha finalitzat, això vol dir que $m = 0$, per tant, per definició de m, podem afirmar que

$$\frac{|v_1 \cdot v_2|}{\|v_1\|^2} \leq \frac{1}{2}. \tag{3.2.1}$$

Sigui $v \in L$ qualsevol vector no nul del reticle. Podem expressar v com

$$v = a_1 v_1 + a_2 v_2 \quad \text{amb } a_1, a_2 \in \mathbb{Z}.$$

Llavor, veiem que

$$\begin{aligned}
\|v\|^2 &= \|a_1v_1 + a_2v_2\|^2 \\
&= a_1^2\|v_1\|^2 + 2a_1a_2(v_1 \cdot v_2) + a_2^2\|v_2\|^2 \\
&\geq a_1^2\|v_1\|^2 - 2|a_1a_2||v_1 \cdot v_2| + a_2^2\|v_2\| \\
&\geq a_1^2\|v_1\|^2 - |a_1a_2|\|v_1\|^2 + a_2^2\|v_2\| && \text{per 3.2.1,} \\
&\geq a_1^2\|v_1\|^2 - |a_1a_2|\|v_1\|^2 + a_2^2\|v_1\| && \text{ja que } \|v_2\| \geq \|v_1\|, \\
&= (a_1^2 - |a_1||a_2| + a_2^2)\|v_1\|^2.
\end{aligned}$$

Desenvolupament aquest coeficient

$$\begin{aligned}
a_1^2 - a_1a_2 + a_2^2 &= \left(a_1 - \frac{1}{2}a_2\right)^2 + \frac{3}{4}a_2^2 \\
&= \frac{3}{4}a_1^2 + \left(\frac{1}{2}a_1 - a_2\right)^2
\end{aligned}$$

veiem que només s'anul·la quan $a_1 = a_2 = 0$. Però com que a_1 i a_2 són enters i no simultàniament nuls, ja que v és un vector no nul, deduïm que $\|v\|^2 \geq \|v_1\|^2$. Així, queda demostrat que v_1 és un dels vectors no nuls més curts del reticle L. □

3.2.2 LLL: l'algorisme de reducció de reticles

Hem vist que l'algorisme de Gauss resol eficientment el problema SVP en dimensió 2. Ara, és natural preguntar-se si existeixen algorismes que resolen el problema en dimensions superiors. Encara que és intuïtiu intentar generalitzar l'algorisme de Gauss per dimensions més grans, això pot portar a diversos problemes. Un exemple seria, a l'hora d'escollir la combinació lineal corresponent per reduir v_n a partir de $v_1, \cdots, v_{n-1}$ estaríem resolent CVP per un subreticle, i aquest es tracta d'un problema complicat. Es pot trobar una explicació més detallada a l'article de Nguyen i Stehlé [4].

En resposta a aquest problema A.K. Lenstra, H.W. Lenstra i L. Lovász van publicar l'any 1982 l'algorisme LLL [5], un algorisme que permet reduir bases de reticles en temps polinomial.

Abans d'introduir l'algorisme LLL, estudiarem una sèrie de definicions i proposicions que ens ajudaran a entendre millor l'algorisme.

Donat una base $\{v_1, v_2, \cdots, v_n\}$ del reticle L, l'objectiu és fer certes modificacions per tal d'obtenir una millor base. És a dir, volem escurçar els vectors el màxim possible. Alternativament, volem que la nova base sigui el més ortogonal possible, és a dir, que el $v_i \cdot v_j$ sigui el més proper possible a zero, per tot i, j.

Apliquem l'algorisme de Gram-Schmidt, tal i com indica el Teorema 2.1.1 per crear una base ortogonal a partir de la base inicial. Així, tenim

$$\begin{aligned}
v_1^* &= v_1 \\
v_i^* &= v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^*, \quad \text{on } |\mu_{i,j}| = \frac{v_i \cdot v_j^*}{\|v_j^*\|^2} \quad \text{per } 1 \leq j \leq i-1.
\end{aligned}$$

La base obtinguda $\mathcal{B}^* = \{v_1^*, v_2^*, \cdots, v_n^*\}$ és una base ortogonal de l'espai vectorial format per la base original $\mathcal{B} = \{v_1, v_2, \cdots, v_n\}$. Però, ens adonem que $\mathcal{B}^*$ no és una

base del reticle L ja que l'algorisme de Gram-Schmidt realitza combinacions lineals amb coeficients no enters dels vectors. En canvi, la proposició següent demostra que les dues bases tenen el mateix determinant.

Proposició 3.2.1. *Sigui $\mathcal{B} = \{v_1, v_2, \cdots, v_n\}$ una base del reticle L i $\mathcal{B}^* = \{v_1^*, v_2^*, \cdots, v_n^*\}$ la base ortogonal de Gram-Schmidt associada. Es compleix*

$$\det(L) = \prod_{i=1}^{n} \|v_i^*\|$$

Demostració. Sigui $F = F(v_1, \cdots, v_n)$ la matriu que té per files els vectors de la base del reticle L $\{v_1, \cdots, v_n\}$. La Proposició 2.2.10 ens diu que $\det(L) = |\det(F)|$. Anàlogament, $F^* = F(v_1^*, \cdots, v_n^*)$ és la matriu que té per files els vectors de la base $\{v_1^*, \cdots, v_n^*\}$.

Pel Teorema 2.1.1, sabem que les dues matrius estan relacionades per

$$MF^* = F$$

on M és la matriu de canvi de base:

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \mu_{2,1} & 1 & 0 & \cdots & 0 & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{n-1,1} & \mu_{n-1,2} & \mu_{n-1,3} & \cdots & 1 & 0 \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \cdots & \mu_{n,n-1} & 1 \end{pmatrix}, \tag{3.2.2}$$

on $|\mu_{i,j}| = \frac{|v_i \cdot v_j^*|}{\|v_j^*\|^2} \leq \frac{1}{2}$.

Com que M és una matriu triangular inferior amb 1 a la diagonal principal, sabem que $\det(M) = 1$. Per tant

$$\det(L) = |\det(F)| = |\det(MF^*)| = |\det(\mathrm{M}) \cdot \det(F^*)| = |\det(F^*)| = \prod_{i=1}^{n} \|v_i^*\|.$$

Aquesta darrera igualtat és conseqüència del fet que els vectors fila v_i^* de la matriu F^* són ortogonals dos a dos. És a dir,

$$\begin{aligned} \det(F^*)^2 &= \det(F^* \cdot (F^*)^t) \\ &= \det\left(\begin{pmatrix} v_1^* \\ v_2^* \\ \vdots \\ v_n^* \end{pmatrix} \begin{pmatrix} v_1^* & v_2^* & \cdots & v_n^* \end{pmatrix} \right) \\ &= \det\begin{pmatrix} \|v_1^*\|^2 & 0 & \cdots & 0 & 0 \\ 0 & \|v_2^*\|^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \|v_{n-1}^*\|^2 & 0 \\ 0 & 0 & \cdots & 0 & \|v_n^*\|^2 \end{pmatrix} \quad \mathcal{B}^* \text{ és una base ortogonal} \\ &= \prod_{i=1}^{n} \|v_i^*\|^2. \end{aligned}$$

Per tant, fent l'arrel ens queda que $|\det(F^*)| = \prod_{i=1}^{n} \|v_i^*\|$. □

Definició 3.2.2. *Sigui V un espai vectorial i $W \subset V$ un subespai vectorial de V. El complement ortogonal de W (en V) és*

$$W^{\perp} = \{v \in V : v \cdot w = 0, \text{ per a tot } w \in W\}.$$

No és difícil veure que $W^{\perp}$ és també un subespai vectorial de V i que qualsevol vector $v \in V$ es pot expressar com $v = w + w'$, on $w \in W$ i $w' \in W^{\perp}$ són únics.

Un cop vist aquesta definició, podem descriure la construcció de Gram-Schmidt com

$$v_i^* = \text{Projecció de } v_i \text{ sobre } \langle v_1, \ldots, v_{i-1}\rangle^{\perp}.$$

Tal i com hem esmentat anteriorment, $\mathcal{B}^* = \{v_1^*, v_2^*, \cdots, v_n^*\}$ no és una base del reticle L. Tot i així, té una funció important a l'hora de definir conceptes claus per l'algorisme LLL.

Definició 3.2.3. *Sigui $\mathcal{B} = \{v_1, v_2, \cdots, v_n\}$ una base del reticle L i $\mathcal{B}^* = \{v_1^*, v_2^*, \cdots, v_n^*\}$ la base ortogonal de Gram-Schmidt associada. Diem que $\mathcal{B}$ és una base LLL reduïda si compleix les dues condicions:*

$$(\textbf{\textit{Condició de mida}}) \qquad |\mu_{i,j}| = \frac{|v_i \cdot v_j^*|}{\|v_j^*\|^2} \leq \frac{1}{2} \quad \text{per a tot } 1 \leq j < i \leq n$$

$$(\textbf{\textit{Condició de Lovász}}) \quad \|v_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)\|v_{i-1}^*\|^2 \qquad \text{per a tot } 1 < i \leq n$$

La condició de Lovász també es pot expressar de formes alternatives. Per exemple,

$$\|v_i^* + \mu_{i,i-1}v_{i-1}^*\|^2 \geq \frac{3}{4}\|v_{i-1}^*\|^2,$$

que també és equivalent a la desigualtat

$$\|\textit{Projecció de } v_i \textit{ sobre } \langle v_1, \ldots, v_{i-2}\rangle^{\perp}\| \geq \frac{3}{4}\|\textit{Projecció de } v_{i-1} \textit{ sobre } \langle v_1, \ldots, v_{i-2}\rangle^{\perp}\|.$$

El resultats principals als que van arribar Lenstra, Lenstra i Lovász van ser:

- Una base LLL reduïda és una bona base.
- Podem trobar una base LLL reduïda en un temps polinomial en n, la dimensió del reticle L.

A continuació veurem les propietats que fan que una base LLL reduïda sigui considerada bona i finalment introduïrem l'algorisme LLL.

Teorema 3.2.4. *Sigui L un reticle de dimensió n. Qualsevol base LLL reduïda $\{v_1, v_2, \cdots, v_n\}$ de L satisfà les dues propietats:*

1. $\prod_{i=1}^{n} \|v_i\| \leq 2^{n(n-1)/4} \cdot \det L$,
2. $\|v_j\| \leq 2^{(i-1)/2}\|v_i^*\| \qquad \text{per a tot } 1 \leq j \leq i \leq n.$

A més, el primer vector d'una base LLL reduïda compleix:

1. $\|v_1\| \leq 2^{(n-1)/4} \cdot |\det L|^{1/n}$,
2. $\|v_1\| \leq 2^{(n-1)/2} \min_{0 \neq v \in L} \|v\|$.

Demostració. Combinant la condició de Lovász i la desigualtat $|\mu_{i,i-1}| \leq \frac{1}{2}$, obtenim:

$$\|v_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|v_{i-1}^*\|^2 \geq \left(\frac{3}{4} - \frac{1}{4}\right) \|v_{i-1}^*\|^2 = \frac{1}{2}\|v_{i-1}^*\|^2.$$

Aplicant repetitivament asquesta desigualtat obtenim

$$\|v_j^*\|^2 \leq 2^{i-j}\|v_i^*\|^2. \tag{3.2.3}$$

1.

$$\begin{aligned}
\|v_i\|^2 &= \left\| v_i^* + \sum_{j=1}^{i-1} \mu_{i,j} v_j^* \right\|^2 && \text{per la construcció de Gram-Schmidt,} \\
&= \|v_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|v_j^*\|^2 && \text{ja que } v_1^*, \cdots, v_n^* \text{és una base ortogonal,} \\
&\leq \|v_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4}\|v_j^*\|^2 && \text{ja que } |\mu_{i,j}| \leq \frac{1}{2}, \\
&\leq \|v_i^*\|^2 + \sum_{j=1}^{i-1} 2^{i-j-2}\|v_i^*\|^2 && \text{per (3.2.3),} \\
&= \left(1 + \sum_{j=1}^{i-1} 2^{i-j-2}\right) \cdot \|v_i^*\|^2 \\
&= \frac{1 + 2^{i-1}}{2}\|v_i^*\|^2 && \text{ja que } 2^{i-3} + \cdots + 2 + 1 = 2^{i-2} - 1 \\
&\leq 2^{i-1}\|v_i^*\|^2 && \text{ja que } 1 \leq 2^{i-1} \text{per a tot } i \geq 1.
\end{aligned}$$

Havent desenvolupat aquesta desigualtat, podem fer el productori de $\|v_i\|^2$ i obtenim

$$\prod_{i=1}^{n} \|v_i\|^2 \leq \prod_{i=1}^{n} 2^{i-1}\|v_i^*\|^2 = 2^{\sum_{i=0}^{n-1} i} \prod_{i=1}^{n} \|v_i^*\|^2 = 2^{n(n-1)/2} \prod_{i=1}^{n} \|v_i^*\|^2 = 2^{n(n-1)/2}\det(L)^2.$$

Aquesta última igualtat s'obté aplicant la Proposició 3.2.1. Ara, aplicant l'arrel obtenim

$$\prod_{i=1}^{n} \|v_i\| \leq 2^{n(n-1)/4} \cdot \det L.$$

2. Per a demostrar la segona propietat primer apliquem la desigualtat provada a la propietat anterior

$$\|v_i\|^2 \leq 2^{i-1}\|v_i^*\|^2$$

i seguidament apliquem (3.2.3). Per tant, per a tot $j \leq i$

$$\|v_j\|^2 \leq 2^{j-1}\|v_j^*\|^2 \leq 2^{j-1} \cdot 2^{i-j}\|v_i^*\|^2 = 2^{i-1}\|v_i^*\|^2.$$

Aplicant l'arrel haurem provat que

$$\|v_j\| \leq 2^{(i-1)/2}\|v_i^*\| \quad \text{per a tot } 1 \leq j \leq i \leq n.$$

Ara pasem a demostrar les propietats que compleix el primer vector d'una base LLL reduïda.

1. Apliquem la segona propietat per $j = 1$ i utilitzem la Proposició 3.2.1 per obtenir

$$\|v_1\|^n \leq \prod_{i=1}^{n} 2^{(i-1)/2}\|v_i^*\| = 2^{\sum_{i=1}^{n}(i-1)/2}\prod_{i=1}^{n}\|v_i^*\| = 2^{\frac{1}{2}\sum_{i=0}^{n-1} i}\prod_{i=1}^{n}\|v_i^*\|$$
$$= 2^{n(n-1)/4}\prod_{i=1}^{n}\|v_i^*\| = 2^{n(n-1)/4}\cdot \det L.$$

   Aplicant l'arrel n-èssima trobem que

$$\|v_1\| \leq 2^{(n-1)/4}\cdot(\det L)^{1/n}.$$

2. Sigui $v \in L$ un vector no nul del reticle, podem expressar v de la següent forma

$$v = \sum_{j=1}^{i} a_j v_j = \sum_{j=1}^{i} b_j v_j^*$$

   on $a_1, \ldots, a_i \in \mathbb{Z}$ amb $a_i \neq 0$, per tant $|a_i| \geq 1$, i $b_1, \ldots, b_i \in \mathbb{R}$. Per deficinió, sabem que per a qualsevol k, els vectors $v_1^*, \ldots, v_k^*$ són ortogonals dos a dos. A més, l'espai vectorial format per aquests vectors coincideix amb l'espai vectorials format per els vectors $v_1, \ldots, v_k$, tal i com hem demostrat al Teorema de Gram-Schmidt 2.1.1. Llavors,

$$v \cdot v_i^* = a_i v_i \cdot v_i^* = b_i v_i^* \cdot v_i^* \quad \text{i} \quad v_i \cdot v_i^* = v_i^* \cdot v_i^*.$$

   Per tant, podem concloure que $a_i = b_i$. En conseqüència, $|b_i| = |a_i| \geq 1$. Aplicant aquesta darrera desigualtat i la segona propietat ja demostrada, per $j = 1$ tenim que

$$\|v\|^2 = \sum_{j=1}^{i} b_j^2\|v_j^*\|^2 \geq b_i^2\|v_i^*\|^2 \geq \|v_i^*\|^2 \geq 2^{-(i-1)}\|v_1\|^2 \geq 2^{-(n-1)}\|v_1\|^2.$$

   Finalment, aplicant l'arrel quadrada obtenim

$$\|v\| \geq 2^{-(n-1)/2}\|v_1\|.$$

   Per tant, sabent que v és qualsevol vector no nul del reticle L, arreglant la desigualtat provem que

$$\|v_1\| \leq 2^{(n-1)/2}\min_{0\neq v\in L}\|v\|.$$

□

Un cop vistes totes aquestes propietats, ja podem introduïr l'algorisme LLL. En aquest algorisme hi ha dues condicions que cal tenir en compte, la condició de mida i la condició de Lovász. Donada una base $\{v_1, v_2, \ldots, v_n\}$ és senzill crear una nova base que satisfà la condició de mida, simplement restem a cada vector v_k de la base original un múltiple enter dels vectors anteriors $v_1, \ldots, v_{k-1}$, tal i com feiem en dimensió 2 a l'algorisme de Gauss (3.2.1). Però a l'algorisme LLL la nova base també ha de satisfer la condició de Lovász, per tant, després de fer la reducció de mida de cada vector ens assegurarem que se satisfà la condició de Lovász. En cas de no complir la condició, reordenem els vectors i els tornem a reduïr.

Teorema 3.2.5. *(**Algorisme LLL**) Sigui $\{v_1, \ldots, v_n\}$ una base del reticle L i sigui $\{v_1^*, \ldots, v_n^*\}$ la base ortogonal de Gram-Schmidt associada. El següent algorisme acaba amb un nombre finit de passos i retorna una base LLL reduïda de L.*

Algorithm 4 Algorisme LLL

```
Introduir {v_1, ..., v_n} una base del reticle L
k = 2
v_1^* = v_1
while k ≤ n do
    for j = k − 1, k − 2, ..., 1, 0 do
        v_k = v_k − ⌊μ_{k,j}⌉ v_j                          ▷ Reducció de mida
        Actualitzar las base de Gram-Schmidt associada
    end for
    if ‖v_k^*‖^2 ≥ (3/4 − μ_{k,k−1}^2) ‖v_{k−1}^*‖^2 then   ▷ Condició de Lóvasz
        k = k + 1
    else
        w = v_{k−1}                                        ▷ Intercanviar v_{k−1} i v_k
        v_{k−1} = v_k
        v_k = w
        Actualitzar las base de Gram-Schmidt associada
        k = max(k − 1, 2)
    end if
end while
return {v_1, ..., v_n} base LLL reduïda
```

Nota: $\mu_{i,j} = (v_i \cdot v_j^*)/\|v_j^*\|^2$.

Sigui $B = \max \|v_i\|$, l'algorisme executarà el while principal un màxim de $\mathcal{O}(n^2 \log n + n^2 \log B)$ vegades. En particular, l'algorisme LLL s'executa en un temps polinomial.

A simple vista pot semblar que l'algorisme LLL és una generalització n-dimensional de l'algorisme de Gauss (3.2.1). Però en realitat existeixen dues diferències crucials que són essencials per demostrar que l'algorisme LLL s'executa en un temps polinomial.
Per un costat, l'algorisme de Gauss calcula el valor mínim per $\|v_2 + qv_1\|$ on $q \in \mathbb{Z}$. En canvi, en LLL el coeficient $\mu_{k,j}$ depèn de v_k i v_j^*, per tant no necessàriament minimitza $\|v_k\|$. Clarament, aplicant l'algorisme LLL en dimensió 2, el coeficient $\mu_{2,1}$ coincideix amb el valor utilitzat a l'algorisme de Gauss, per tant els passos de reducció de mida seran iguals.
Per l'altre costat, a l'algorisme LLL el control de mida que es realitza a través de la condició de Lovász es fa sobre la base ortogonal de Gram-Schmidt, mentre que a l'algorisme de Gauss el control es fa sobre la mida dels vectors de la base original.

Per implementar l'algorisme LLL de manera eficient ens podem trobar amb diverses dificultats. Una d'elles és la que acabem d'esmentar, la condició de Lovász, on s'utilitza la base ortogonal de Gram-Schmidt i els coeficients $\mu_{i,j}$. En una implementació eficient hauríem de calcular aquests valors i guardar-los per utilitzar-los més endavant. No hem inclós aquesta part a l'algorisme anterior ja que no és rellevant per entendre l'algorisme LLL i tampoc per demostrar que l'algorisme s'executa en temps polinomial. Una altra dificultat es dona quan un vol implementar la reducció LLL en un reticle enter amb valors exactes ja que els càlculs poden involucrar nombres molt grans. En general, per treballar

amb reticles de dimensions elevades és necessari fer arrodoniments de nombre decimals que poden comportar a problemes d'errors. No ens adentrarem en detalls amb aquestes dificultats però el lector les hauria de tenir present.

Abans de començar amb la demostració del teorema, estudiarem primer la intuïció que hi ha darrere de l'intercanvi que s'executa quan v_k no compleix la condició de Lovász (Pas 11).

El principal objectiu de l'algorisme LLL és executar una sèrie de vectors curts ordenats de petit a gran. Per a cada $1 \leq l \leq n$, sigui L_l el reticle format per la base $v_1, \ldots, v_l$, ens adonem que al llarg de l'algorisme, L_l va canviant, només L_n es manté igual ja que és el reticle sencer. Realment és normal que passi aquest fet ja que el que pretèn l'algorisme LLL és ordenar els vectors de la base tal que els determinants $\det(L_l)$ siguin mínims, és a dir, LLL intenta minimitzar els volums dels dominis fonamentals dels subreticles $L_1, \ldots, L_n$.

Sabem que quan no es compleix la condició de Lovász es té que

$$\|\text{Projecció de } v_i \text{ sobre } \langle v_1, \ldots, v_{i-2}\rangle^{\perp}\| < \frac{3}{4}\|\text{Projecció de } v_{i-1} \text{ sobre } \langle v_1, \ldots, v_{i-2}\rangle^{\perp}\|.$$

Ara, si canviem el valor 3/4 per 1 l'algorisme sempre intercanviarà els vector v_k i v_{k-1} reduint així el determinant $\det(L_{k-1})$. Desafortunadament, en aquest cas seria un problema a investigar si l'algorisme finalitza o no en temps polinomial.

En canvi, en cas d'utilitzar el valor 3/4 o qualsevol altre estricament menor que 1, l'algorisme acaba en temps polinomial, però podria ser que deixés de fer alguna de les reduccions. Per exemple, en el primer pas, intercanviem els vectors només si $\|v_2\| < \frac{3}{4}\|v_1\|$, mentre que es podria reduir el determinant si imposem que es fes l'intercanvi de vectors sempre que $\|v_2\| < \|v_1\|$. A la pràctica, hi ha més preferència per agafar una constant entre 3/4 i 1 per la condició de Lovász.

Observem que l'efecte d'intercanviar els vectors v_k i v_{k-1} és fer el coeficient $\mu_{k,k-1}$ més gran. Això ens permet reduir el nou vector v_k utilitzant el nou vector v_{k-1} i fent-los més ortogonals entre ells.

Demostració. (del Teorema 3.2.5) Per el cas que necessitem i per simplificar desenvoluparem la demostració per a $L \subset \mathbb{Z}^n$ un reticle enter.

Queda clar que si l'algorisme LLL acaba llavors retorna una base LLL reduïda, ja que el *For* (passos 5-7) garanteix que es compleix la condició de mida i el fet que $k = n + 1$ al finalitzar l'algorisme assegura que tots els vectors de la base compleixen la condició de Lovász.

Per demostrar que l'algorisme finalitza en un temps finit provarem que el pas 14 només s'executa un nombre finit de vegades. D'aquesta manera es garanteix que el valor del comptador k sempre acabarà sobrepassant n finalitzant així el bucle principal i acabant l'algorisme.

Sigui $\{v_1, \ldots, v_n\}$ una base del reticle L i $v_1^*, \ldots, v_n^*$ la base ortogonal de Gram-Schmidt associada. Sigui L_l els subreticles

$$L_l = \text{subreticle generat per } \{v_1, \ldots, v_l\} \quad \text{per a cada } l = 1, 2, \ldots, n.$$

Definim les quantitats

$$d_l = \prod_{i=1}^{l} \|v_i^*\|^2 \quad \text{i} \quad D = \prod_{l=1}^{n} d_l = \prod_{i=1}^{n} \|v_i^*\|^{2(n+1+i)}.$$

Fent ús d'un argument similar a la demostració de la Proposició 3.2.1, veiem que

$$\det(L_l)^2 = d_l. \tag{3.2.4}$$

Veiem que el valor de D només varia quan s'executa l'intercanvi de vectors (passos 11-13), és més, quan això passa l'únic d_l que varia és quan $l = k-1$. Si $l < k-1$, llavors a d_l no s'inclouen v_{k-1}^* i v_k^*; si $l \geq k$ llavors els producte inclourà tant v_{k-1}^* com v_k^*, per tant encara que s'intercanviïn els vectors el producte romandrà el mateix.

Si s'efectua l'intercanvi de vectors vol dir que la condició de Lovász no se satisfà, per tant

$$\|v_k^*\|^2 < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|v_{k-1}^*\|^2 \leq \frac{3}{4}\|v_{k-1}^*\|^2. \tag{3.2.5}$$

Per tant, intercanviar v_{k-1}^* amb v_k^* té el següent efecte sobre valor de d_{k-1}:

$$\begin{aligned} d_{k-1}^{nou} &= \|v_1^*\|^2 \cdot \|v_2^*\|^2 \cdots \|v_{k-2}^*\|^2 \cdot \|v_k^*\|^2 \\ &= \|v_i^*\|^2 \cdot \|v_2^*\|^2 \cdots \|v_{k-2}^*\|^2 \cdot \|v_{k-1}^*\|^2 \cdot \frac{\|v_k^*\|^2}{\|v_{k-1}^*\|^2} \\ &= d_{k-1}^{antic} \cdot \frac{\|v_k^*\|^2}{\|v_{k-1}^*\|^2} \\ &\leq \frac{3}{4} \cdot d_{k-1}^{antic}, \qquad \text{per 3.2.5} \end{aligned}$$

on d_{k-1}^{nou} és la quantitat corresponent després de fer l'intercanvi de vectors i d_{k-1}^{antic} el d'abans de l'intercanvi.
Així, si l'intercanvi de vectors s'efectua N vegades el valor de D s'estaria reduint al menys $(3/4)^N$ vegades, ja que per cada intercanvi d_l es redueix en un factor de $\frac{3}{4}$.

Com hem assumit que $L \subset \mathbb{Z}^n$ podem afirmar que tot vector no nul de L tindrà un mòdul més gran o igual que 1. Per tant, aplicant el teorema d'Hermite (Teorema 2.3.2) al subreticle L_l obtenim

$$1 \leq \min_{0 \neq w \in L_l} \|w\| \leq \sqrt{l} \cdot \det(L_l)^{1/l}.$$

Elevant al quadrat obtenim que

$$1 \leq l \cdot \det(L_l)^{2/l},$$

per tant,

$$l^{-l} \leq \det(L_l)^2.$$

Ara, utilitzant aquesta desigualtat i (3.2.4) obtenim

$$d_l = \det(L_l)^2 \geq l^{-l}.$$

A continuació, fem el productori per trobar una fita inferior de D

$$D = \prod_{l=1}^{n} d_l \geq \prod_{l=1}^{n} l^{-l} \geq \prod_{l=1}^{n} l^{-n} = (n!)^{-n} \geq n^{-n^2}.$$

Aquesta darrera desigualtat és deguda a que $n! \leq n^n$.
Hem vist que D està fitat inferiorment per un valor diferent de zero que només depèn de la dimensió del reticle L. Per tant només es podrà reduïr en un factor de $3/4$ un nombre finit de vegades. Així, podem afirmar que l'intercanvi de vectors s'efectuarà un nombre finit de vegades, és a dir, l'algorisme LLL finalitza.

Ara anem a provar que l'algorisme s'executa en un temps polinomial. Sigui $D_{inicial}$ el valor de D corresponen a la base original, sigui D_{final} el valor de D de la base al finalitzar l'algorisme i N els nombre de vegades que s'executa l'intercanvi de vectors, podem aplicar els resultats anteriors per trobar que

$$n^{-n^2} \leq D_{final} \leq \left(\frac{3}{4}\right)^N D_{inicial}.$$

Observem que el bucle principal s'executarà un total de $2N + n$ vegades, per tant per trobar un màxim d'execucions és suficient amb trobar una fita superior de N. Prenent el logaritme de la desigualtat anterior tenim que

$$-n^2 \log n \leq N \log \frac{3}{4} + \log D_{inicial}.$$

Notem que $\log(3/4) < 1$, és a dir, $-\log(3/4) > -1$. Per tant,

$$N < -N \log \frac{3}{4} \leq n^2 \log n + \log D_{inicial}.$$

Finalment obtenim

$$N = \mathcal{O}(n^2 \log n + \log D_{inicial}).$$

Per a acabar la demostració estimem el valor de $D_{inicial}$. Per la construcció de Gram-Schmidt sabem que $\|v_i^*\| \leq \|v_i\|$, per tant

$$\begin{aligned} D_{inicial} &= \prod_{i=1}^{n} \|v_i^*\|^{2(n+1-i)} \leq \prod_{i=1}^{n} \|v_i\|^{2(n+1-i)} \leq \max_{1\leq i\leq n} \|v_i\|^{\sum_{i=1}^{n} 2(n+1-i)} \\ &= \max_{1\leq i\leq n} \|v_i\|^{2\sum_{i=1}^{n} i} = \max_{1\leq i\leq n} \|v_i\|^{n^2+n} = B^{n^2+n}, \end{aligned}$$

on $B = \max \|v_i\|$. Per tant, prenent el logaritme tenim que

$$\log D_{inicial} = \mathcal{O}(n^2 \log B).$$

Així hem demostrat que el bucle principal de l'algorisme s'executarà un màxim de $\mathcal{O}(n^2 \log n + n^2 \log B)$ vegades. □

Com hem mencionat al principi de la secció, l'algorisme LLL destaca per ser una possible amenaça per la seguretat dels criptosistemes basats en reticles. Sabem que si el reticle L té una base ortogonal resoldre els problemes SVP i CVP resulta molt senzill. Encara que l'algorisme LLL no retorna una base ortogonal, aquesta és quasi-ortogonal, és a dir, el producte escalar entre vectors és molt proper a 0. Així, combinant l'algorisme LLL amb l'algorisme de Babai (3.1) podem arribar a resoldre apprCVP.

Teorema 3.2.6. *(**Algorisme LLL per apprCVP**) Sigui L un reticle qualsevol de dimensió n donada per la base $v_1, \ldots, v_n$, existeix una constant C tal que el següent algorisme resol apprCVP en C^n passos.*

Algorithm 5 Algorisme LLL i Babai per apprCVP

$v_1, \ldots, v_n$ base del reticle L.
Aplicar l'algorisme LLL sobre $v_1, \ldots, v_n$.
Aplicar l'algorisme de Babai sobre la base LLL reduïda obtinguda.
return v ▷ vector resultant d'aplicar l'algorisme de Babai a la base LLL reduïda

Demostració. No realitzarem la demostració ja que no és essencial per el desenvolupament d'aquest treball. □

Capítol 4

Criptosistemes basats en reticles

4.1 Introducció

A mitjans dels anys 90, es van introduir criptosistemes basats en el problema del vector més curt SVP i el problema del vector més proper CVP en un reticle L de dimensió n. Els criptosistemes que més van destacar van ser el criptosistema d'AjtaiDwork [13], el cripstosistema GGH de Goldreich, Goldwasser i Halevi [14], i el criptosistema NTRU de Hoffstein, Pipher i Silverman [15].

L'origen d'aquests criptosistemes va ser motivat per dues causes principals. Per una banda, es volia optimitzar els criptosistemes anteriors. Els criptosistemes basats en reticles són més ràpids que els cripstosistemes basats en la factorització o logaritmes discrets, com són el RSA, ElGamal, ECC, entre d'altres. Encriptacions i desencriptacions dels sistemes RSA, ElGamal i ECC requereixen $\mathcal{O}(k^3)$ operacions, on k són els bits de seguretat que volem aconseguir, mentre que per sistemes basats en reticles només es requereixen $\mathcal{O}(k^2)$ operacions. Per l'altra banda, sempre és d'interés crear criptosistemes basats en una sèrie de problemes matemàtics difícils per millorar el nivell de seguretat dels criptosistemes.

Anys enrere, els criptosistemes basats en reticles encara no eren tan ben apreciats com els criptosistemes basats en la factorització o logaritmes discrets. És per això que la implementació d'aquests criptosistemes era molt petita comparada amb la dels antics criptosistemes. Avui dia, l'arribada dels ordinadors quàntics i la seva ràpida evolució podria significar la fi dels criptosistemes ja existents. En conseqüència la importància de la criptografia postquàntica ha crescut exponencialment i cada cop hi ha més recerques sobre criptosistemes postquàntics. Per tant, al ser una bona candidata com a criptosistemes resistents a atacs quàntics, la criptografia basada en reticles també ha rebut una especial importància.

En aquest treball estudiarem detalladament el criptosistema GGH, un criptosistema basat en reticles que implementa directament les idees esmentades en el capítol anterior.

Un petit resum del criptosistema GGH seria el següent:
Suposem que l'Alice té una bona base $\mathcal{B}_{good}$ d'un reticle L com a clau privada i una base dolenta $\mathcal{B}_{bad}$ com a clau pública. El missatge que vol transmetre en Bob és un vector binari m, i calcula la combinació lineal de m amb els vectors de la base $\mathcal{B}_{bad}$

$$v = \sum_i m_i v_i^{bad}.$$

Finalment, li afegeix un vector aleatori r amb coordenades petites.
Com que l'Alice coneix una bona base de L, $\mathcal{B}_{good}$, implementant l'algoritme de Babai troba v. Així pot recuperar el missatge m expressant v en funció dels vectors de la base $\mathcal{B}_{bad}$. En canvi, l'Eve desconeix la base $\mathcal{B}_{good}$ per tant no pot resoldre el problema del vector més proper CVP en L.

Una clau pública del criptosistema GGH consisteix en n^2 nombres. Inicialment, el tamany d'una clau era de $\mathcal{O}(n^3 \log n)$, més tard, aplicant la idea de Micciancio, es va poder reduir a $\mathcal{O}(n^2 \log n)$.

Goldreich, Goldwasser i Halevi van conjecturar que per $n > 300$, el problema del vector més proper CVP seria impossible de resoldre. Aquest afectava directament a l'aplicació de l'algorisme GGH ja que també seria intractable per dimensions superiors a 300. En resposta a aquest problema, Phong Nguyen va provar que una transformació del criptosistema GGH reduïa el problema del vector més proper CVP a una versió més senzilla. Aquest li va poder permetre aplicar el criptosistema GGH en dimensions superiors a 350.

Un dels primers atacs contra el criptosistema GGH més destacats va ser presentat per Phong Nguyen l'any 1999. Nguyen va provar que per tal de garantir la seguretat del criptosistema, s'havia de treballar amb reticles de dimensions superiors a 350. (Es pot consultar [8] per explicacions amb detall.)
Anys després de l'atac de Nguyen, Han, Daewan, Kim Myung Han i Yeom Yongjin van presentar una nova variant del criptosistema GGH, el criptosistema Paeng-Jung-Ha o també conegut com el crptosistema PJH. Juntament amb aquesta nova variant van presentar també un altre atac contra el GGH i van demostrar que en certes situacions especials el criptosistema deixava de ser segur fins a dimensions tant grans com 1000. (Per més detalls es pot consultar [9].) Tot i així, desxifrar el missatge correctament i obtenir el text original depenia d'aquestes situacions especials, per tant, no es podien aplicar a casos generals.

És cert que la introducció dels algorismes de reducció de reticles, com és l'algorisme LLL, suposa una gran amenaça contra la seguretat del criptosistema GGH. Però en dimensions molt elevades aquests algorismes passen a ser impracticables. Malauradament, cal esmentar que en dimensions molt elevades les claus generades pel criptosistema GGH també es fan molt grans i difícils de tractar. Tot i així, el criptosistema GGH encara es considera un dels més competitius criptosistemes basats en reticles.

4.2 El criptosistema de clau pública GGH

Un cop introduït els criptosistemes basats en reticles, anem a estudiar detalladament el criptosistema GGH.

Per començar l'Alice escull un conjunt de vectors linealment independents

$$v_1, v_2, \ldots, v_n \in \mathbb{Z}^n$$

bastant ortogonals entre ells per crear la seva clau privada. Una forma de trobar vectors linealment independents és fixant un paràmetre d i triar les coordenades de $v_1, v_2, \ldots, v_n$ aleatòriament entre $-d$ i d. Calculant el quocient de Hadamard de la base escollida, l'Alice pot comprobar que efectivament es tracta d'una bona base. Aquesta base serà la seva clau privada. Per conveniència, anomenem V a la matriu $n \times n$ que té per files les coordenades dels vectors de la base, i anomenem L al reticle generat per aquesta base.

A continuació, l'Alice escull una matriu $n \times n$, U, amb coeficients enters i det $= \pm 1$. I calcula

$$W = UV.$$

Els vectors files de W, $w_1, w_2, \ldots, w_n$ formaran la nova base de L, que pasarà a ser la clau pública.

1. **Procés de xifratge**
   Per enviar un missatge a l'Alice, en Bob haurà d'escollir un vector m amb valors petits, un exemple podria ser el vector binari. A més, haurà d'escollir un vector aleatori r que actuarà com a clau efímera i haurà de tenir també valors molt petits. Per exemple, sigui $\delta > 0$ un paràmetre públic fix, les coordenades del vector r poden ser valors aleatòris entre $-\delta$ i δ.
   Un cop escollit m i r, haurà de calcular

   $$e = mW + r = \sum_{i=1}^{n} m_i w_i + r,$$

   que serà el seu text xifrat.

2. **Procés de desxifratge**
   Per desxifrar el text, l'Alice haurà d'implementar l'algorisme de Babai sobre la seva clau privada, la base $\{v_1, v_2, \ldots, v_n\}$, per trobar un vector de L proper a e. Com l'Alice treballa amb una bona base i r és petit, l'algorisme de Babai li retornarà mW. Finalment, multiplicant per la matriu inversa de W, recuperarà el missatge orinigal m.

3. **Prova de seguretat**
   Suposem que una tercera persona, l'Eve, que disposa del text xifrat e i la clau pública, vol desxifrar el missatge. Sabem que la clau pública es tracta d'una base dolenta, per tant, a l'hora d'aplicar l'algorisme de Babai, el vector que trobarà serà ben lluny del vector del reticle més proper a e. Per tant, l'Eve obtindrà un text m' diferent de l'original. Així, el missatge original romandrà segur.

Per entendre millor el criptosistema podeu veure el següent esquema, on s'indica de manera clara el passos que s'ha de seguir i l'ordre en el que s'ha de realitzar.

GGH

- 1. Escollir una bona base $\{v_1, \ldots, v_n\}$.
- 2. Escollir una matriu entera U tal que $\det(U) = \pm 1$.
- 3. Calcular $W = UV$ i extreure la base dolenta $\{w_1, \ldots, w_n\}$.
- 4. Fer pública la clau $w_1, \ldots, w_n$

→ **Creació de les claus**

- 5. Escollir el text m a xifrar.
- 6. Escollir un vector aleatori amb valor petits r.
- 7. Utilitzar la clau pública per calcular $e = m_1 w_1 + \ldots + m_n w_n + r$.
- 8. Enviar el text xifrat e a l'Alice.

→ **Procés de xifratge**

- 9. Trobar el vector $v \in L$ més proper a e utilitzant l'algorisme de Babai.
- 10. Calcular el producte vW^{-1} i recuperar m.

→ **Procés de desxifratge**

Presentem un exemple 3-dimensional per veure una aplicació del criptosistema.

Exemple 4.2.1. Sigui L un reticle. L'Alice pren com a clau privada la base bona

$$v_1 = (-10, 8, -10), \quad v_2 = (-10, -14, -14), \quad v_3 = (1, -2, -4).$$

L'Alice calcula el determinant del reticle

$$\det(L) = \begin{vmatrix} -10 & -10 & 1 \\ 8 & -14 & -2 \\ -10 & -14 & -4 \end{vmatrix} = 1052, \tag{4.2.1}$$

i el coeficient de Hadamard

$$\mathcal{H}(v_1, v_2, v_3) = \left(\frac{\det(L)}{\|v_1\|\|v_2\|\|v_3\|} \right)^{1/3} \approx 0.86041$$

i comprova que realment es tracta d'una bona base, ja que el quocient de Hadamard és bastant proper a 1, per tant la base és bastant ortogonal. Llavors, escull la matriu

$$U = \begin{pmatrix} 1 & 322 & -88687 \\ -8325 & -2680649 & 738318668 \\ 25 & 8050 & -2217174 \end{pmatrix}, \tag{4.2.2}$$

que té determinant $\det(U) = 1$, per crear la clau pública w. Aquesta l'extreu per les files de la matriu resultant del producte UV.

$$W = UV = \begin{pmatrix} 1 & 322 & -88687 \\ -8325 & -2680649 & 738318668 \\ 25 & 8050 & -2217174 \end{pmatrix} \cdot \begin{pmatrix} -10 & -10 & 1 \\ 8 & -14 & -2 \\ -10 & -14 & -4 \end{pmatrix}, \tag{4.2.3}$$

$$W = \begin{pmatrix} -91917 & 172874 & 350230 \\ 765208408 & -1439174850 & -2915662336 \\ -2297924 & 4321848 & 8755746 \end{pmatrix}. \tag{4.2.4}$$

Per tant, l'Alice fa pública la base

$$\begin{aligned} w_1 &= (-91917, 172874, 350230), \\ w_1 &= (765208408, -1439174850, -2915662336), \\ w_3 &= (-2297924, 4321848, 8755746). \end{aligned}$$

Anàlogament, l'Alice pot comprovar que w és una base dolenta calculant el seu quocient de Hadamard, que és molt proper a 0.

$$\mathcal{H}(v_1, v_2, v_3) = \left(\frac{\det(L)}{\|w_1\|\|w_2\|\|w_3\|} \right)^{1/3} \approx 0.000000427718.$$

Ara, en Bob li vol enviar a l'Alice el missatge $m = (-46, 78, -100)$ i escull el vector aleatori $r = (-2, -2, -1)$. Amb aquests dos vectors calcula el text xifrat

$$\begin{aligned} e &= (-46, 78, -100) \cdot \begin{pmatrix} -91917 & 172874 & 350230 \\ 765208408 & -1439174850 & -2915662336 \\ -2297924 & 4321848 & 8755746 \end{pmatrix} + (-2, -2, -1) \\ &= (59920276404, -112695775306, -228313347389). \end{aligned}$$

L'Alice, un cop rep el text encriptat, aplica l'algorisme de Babai per desxifrar el text. Tal i com diu l'algorisme, l'Alice comença expressant el vector e amb coeficients reals en funció de la seva base privada,

$$e \approx -651896.017v_1 - 209910433.817v_2 + 57814653105.7v_3.$$

Arrodoneix els coeficients a l'enter més pròxim i troba un vector del reticle proper a e,

$$\begin{aligned} v &= -651896v_1 - 209910434v_2 + 57814653106v_3 \\ &= (59920276406, -112695775304, -228313347388). \end{aligned}$$

Finalment, recupera el missatge m expressant v com a combinació lineal de la base pública, o més especificament dit, multiplica v per la inversa de W,

$$\begin{aligned} vW^{-1} &= (59920276406, -112695775304, -228313347388)\cdot \\ &\quad \cdot \begin{pmatrix} -91917 & 172874 & 350230 \\ 765208408 & -1439174850 & -2915662336 \\ -2297924 & 4321848 & 8755746 \end{pmatrix}^{-1} \\ &= (-46, 78, -100). \end{aligned}$$

Suposem que l'Eve també vol desxifrar el missatge del Bob, però com la clau pública és la base dolenta, l'Eve només coneix $\{w_1, w_2, w_3\}$. Aplicant l'algorisme de Babai a la base pública obtindrà

$$e \approx -1091.21673w_1 + 83.692w_2 + 1837.249w_3.$$

Arrodonint els coeficients als enters més propers, troba el vector del reticle

$$v' = -1091w_1 + 84w_2 + 1837w_3 = (60156501331, -113140058158, -229213431752)$$

Ara si expressem com abans v' com a combinació lineal de la base pública recuperem un text

$$\begin{aligned} m' = v'W^{-1} &= (60156501331, -113140058158, -229213431752)\cdot \\ &\quad \cdot \begin{pmatrix} -91917 & 172874 & 350230 \\ 765208408 & -1439174850 & -2915662336 \\ -2297924 & 4321848 & 8755746 \end{pmatrix}^{-1} \\ &= (-1091, 84, 1837), \end{aligned}$$

que és diferent al vector m original. L'error es podrueix a l'hora d'aplicar l'algorisme de Babai a la base dolenta. Veiem que

$$\|e - v\| \approx 3 \quad \text{mentre que} \quad \|e - v'\| \approx 1031184430.27.$$

Encara que busquem vectors amb nombres molt grans, en dimensió 3 o generalment en dimensió baixa, el criptosistema GGH no és prou segur, ja que existeixen algorismes eficients capaços de trobar una bona base. Els algorismes que hem estudiat en el capítol 3.2 en són exemples.

Observació 4.2.2. Observem que el criptosistema GGH es tracta d'un criptosistema amb caire probabilístics, és a dir, un text pot derivar a diferents texts xifrats segons el vector aleatori r escollit. Aquest fet pot causar greus problemes si en Bob envia el mateix missatge més d'un cop utilitzant vectors r diferents o enviant missatges diferents amb un mateix vector aleatori r.

Observació 4.2.3. Una versió alternativa del criptosistema GGH és intercanviant el missatge m amb el vector r, llavors el text xifrat seria $e = rW + m$. En aquest cas, l'Alice trobaria rW a l'hora de buscar el vector més proper a e i recuperaria el text original calculant $e - rW$.

4.2.1 Aplicació de l'algorisme LLL al criptosistema GGH

L'algorisme LLL (Teorema 3.2.5) té diverses aplicacions en la criptoanàlisi, des d'atacs a criptosistemes de clau pública fins a anàlisi dels criptosistemes més recents basats en reticles. Dins d'aquests criptosistemes en trobem el criptosistema GGH.

En aquest apartat aplicarem l'algorisme LLL sobre l'exemple proposat a l'apartat anterior (Exemple 4.2.1). Som conscients que l'algorisme LLL és capaç de trencar l'exemple degut a que estem treballant en una dimensió molt baixa. A la pràctica, instàncies segures del criptosistema GGH requereix l'ús de reticles de dimensió 500 fins a 1000, que en el cas de GGH comportaria a claus públiques poc pràctiques ja que serien d'un tamany excessiu.

Tal i com hem vist a l'Exemple 4.2.1 la clau pública de l'Alice es tracta de la base dolenta del reticle L,

$$\begin{aligned} w_1 &= (-91917, 172874, 350230), \\ w_1 &= (765208408, -1439174850, -2915662336), \\ w_3 &= (-2297924, 4321848, 8755746). \end{aligned}$$

Per l'altre banda, el missatge encriptat del Bob és

$$e = (59920276404, -112695775306, -228313347389).$$

Ara, l'Eve vol desencriptar el missatge d'en Bob, però ja hem vist que aplicant l'algorisme de Babai a la base pública obté un missatge diferent de l'original. Per tant, aquesta vegada aplicarà primer l'algorisme LLL per trobar una base del reticle L quasi-ortogonal. L'algorisme li retorna la base

$$\begin{aligned} u_1 &= (1, -2, -4), \\ u_2 &= (-14, -6, 2), \\ u_3 &= (-11, 10, -6). \end{aligned}$$

Podem comprovar que realment es tracta d'una bona base calculant el seu quocient de Hadamard.

$$\mathcal{H}(u_1, u_2, u_3) = \left(\frac{\det(L)}{\|u_1\|\|u_2\|\|u_3\|} \right)^{1/3} \approx 0.9768499,$$

on $\det(L)$ és el determinant de la matriu formada pels vectors u_1, u_2, u_3 en fila de la nova base. Observem que aquesta base és fins i tot millor que la base privada de l'Alice, ja que el quocient de Hadamard és encara més proper a 1.
A continuació, l'Eve aplica l'algorisme de Babai (3.1) sobre aquesta nova base i troba el vector

$$u = (59920276406, -112695775304, -228313347388),$$

que és molt proper a e. Finalment, expressa u en funció de la base original, és a dir, la base pública i obté

$$u = -46w_1 + 78w_2 - 100w_3.$$

Així, l'Eve recupera el missatge original d'el Bob $m = (-46, 78, -100)$.

4.3 Altres criptosistemes basats en reticles

Tal com hem comentat anteriorment, el criptosistema GGH va ser un dels primers criptosistemes basats en reticles. És per això que diem que va ser un dels criptosistemes que va introduir la criptografia basada en reticles. A més, molts criptosistemes posteriors s'inspiren o són modificacions d'aquest. Alguns dels que més destaquen són els següents.

El criptosistema de clau pública NTRU va ser proposat per Hoffstein, Piper i Silverman a [15]. Encara que és més natural descriure'l com un criptosistema basat en anells polinomials convolucionals, el problema matemàtic que el sustenta es pot plantejar equivalentment amb la teoria dels reticles en una estructura especial, concretament els problemes SVP (per a la recuperació de la clau) i CVP (per a la recuperació del missatge). El criptosistema consisteix en dos algorismes: NTRUEncrypt, utilitzat pel procés de xifratge i NTRUSign, utilitzat per a realitzar firmes digitals. La clau privada del criptosistema NTRU es tracta d'un vector curt del reticle $(f, g) \in L$. Mentre que la clau pública és una matriu circulant $N \times N$, per tant, té mida $\mathcal{O}(n \log n)$, que és significativament més petit que la clau pública del criptosistema GGH.

El sistema d'Ajtai-Dwork [13] és un criptosistema probabilístic de clau pública i destaca perquè els seus propis autors van demostrar que el criptosistema era segur sempre i quan el pitjor dels casos del problema de reticles u-SVP (de l'anglès *the unique Shortest Vector Problem*) no es pogués solucionar en temps polinomial. En canvi, a la pràctica, la mida de la clau resultava ser d'$\mathcal{O}(n^4)$ i portava a claus excessivament grans. Més tard, Nguyen i Stern van demostrar que qualsevol implementació eficient del sistema d'Ajtai-Dwork era insegura.

L'any 2005, Oded Regev va introduir a [16] el cas general del problema LWE (de l'anglès *Learning With Errors*). Aquest es tracta del problema anàleg aplicat a la criptografia del problema SIS [17] (de l'algès *Short Integer Solution*). Donada una tupla $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ on $\vec{a}$ és un vector escollit uniformement a l'atzar de $\mathbb{Z}_p^*$, $\vec{s}$ un vector donat de $\mathbb{Z}_p^*$, conegut com el *secret*, i e l'error en $\mathbb{Z}_p$, existeixen dues versions principals del problema LWE. Search-LWE: donat un nombre polinomial de mostres de les tuples, trobar el vector secret $\vec{s}$; i Decision-LWE: saber distingir entre mostres amb un vector $\vec{s}$ aleatori i mostres aleatòries distribuïdes uniformement.

Capítol 5

Part pràctica

5.1 Implementació del criptosistema GGH

La part pràctica d'aquest treball està dedicada a la implementació del criptosistema GGH. Fem ús del sistema de software *SageMath* com a plataforma per a desenvolupar el codi amb el llenguatge *python*. A continuació detallarem l'estructura del codi, que consta de 4 parts. (Es pot consultar el codi a l'Apèndix A).

Abans de començar a desenvolupar el codi, caldrà primer establir certs paràmetres, alguns al principi i d'altres durant el codi.

- n = dimensió del reticle L.
- k = paràmetre de criteri del quocient de Hadamard per a determinar si una base és suficientment ortogonal.
- d = paràmetre per a formar la base privada.
- δ = paràmetre per a formar el vector aleatori r.
- m = missatge a encriptar.

La primera part correspon a la creació de les claus. Per crear una bona base fixem un paràmetre d i escollim aleatòriament les coordenades dels vectors de la base en l'interval $[-d, d]$. Comprovem que realment és una bona base calculant i analitzant el seu quocient de Hadamard. Aquesta serà la clau privada. A continuació, per a crear la clau pública, trobem una base dolenta a partir de la base anterior. Volem obtenir una matriu U, tal que a l'operar $W = UV$, els vectors fila de W siguin una base dolenta. Per tant, per a crear la matriu U, realitzarem multiplicacions de matrius elementals fins que el seu quocient de Hadamard sigui molt proper a 0. Sortirem del bucle un cop es trobi una base dolenta.

La segona part del codi es tracta del procés de xifratge. Primer de tot, caldrà introduir el missatge m que es vol xifrar. Llavors, fixem una δ petita qualsevol i triem un vector aleatori r amb coordenades en l'interval $[-\delta, \delta]$. Fem l'operació $e = mW + r$ per obtenir el text xifrat e.

A la tercera part implementem l'algorisme de Babai per a desxifrar e. En primer lloc, expressem e en funció de la base privada, per fer-ho multipliquem e per la inversa de V. Les coordenades obtingudes no tenen perquè ser enteres, per tant, apliquem l'algorisme

de Babai i trobem el vector del reticle L més proper a e. El vector resultant serà mW. Ara, per a recuperar el missatge m només caldrà multiplicar per la inversa de la matriu W.

Finalment, per comprovar que el criptosistema és segur fem una prova de seguretat. Expressem e en funció de la base pública $\{w_1, \ldots, w_n\}$ calculant la inversa de W i multiplicant-la per e. Apliquem l'algorisme de Babai a la base pública i veiem que recuperem un missatge diferent a l'original.

5.2 Implementació de l'algorisme LLL

A part del codi del criptosistema GGH, també hem considerat oportú implementar l'algorisme LLL, ja que es tracta d'una de les majors amenaces del criptosistema. El següent codi és una transcripció directa de l'algorisme LLL. (Podeu consultar el codi a l'Apèndix B)

La primera part del codi es tracta de l'algorisme LLL. Primer de tot calculem la base ortogonal de Gram-Schmidt $\{v_1^*, \ldots, v_n^*\}$ associada a la base pública. To seguit, entrem dins del bucle principal on, a mesura que es va desenvolupant la reducció de mida, comprovem que tots els vectors de la base satisfan la condició de Lovász. Recordem que sempre que modifiquem la base $\{v_1, \ldots, v_n\}$ caldrà actualitzar la base ortogonal de Gram-Schmidt associada. Un cop sortim del bucle, haurem trobat la base LLL reduïda.

A la segona part, apliquem l'algorisme de Babai sobre la base LLL reduïda que acabem d'obtenir. Calculem el seu coeficient de Hadamard per comprovar que realment és una bona base, moltes vegades inclús més ortogonal que la base privada de l'Alice. A continuació, expressem e en funció de la base LLL reduïda i apliquem l'algorisme de Babai per trobar una expressió de e en funció de $\{v_1, \ldots, v_n\}$ amb coeficients enters. El vector resultant serà un dels vectors del reticle més propers a e. Finalment, recuperem el missatge m multiplicant el vector per la inversa de W.

5.3 Resultats experimentals

5.3.1 Criptosistema GGH

Al llarg del treball mencionem diverses vegades el concepte de base bona i base dolenta. Una base bona és una base amb vectors suficientment ortogonals, és a dir, una base que té un quocient de Hadamard molt proper a 1. Malauradament, no sabem amb certesa el valor exacte del quocient a partir del qual es considera una base bona. És per això que dediquem aquest apartat a estudiar els resultats experimentals de la implementació del criptosistema GGH per diferents paràmetres de k, el paràmetre de criteri del quocient de Hadamard per a determinar si una base és suficientment ortogonal o no.

$k = 0.5$, $n = 3$, $m = (-46, 78, -100)$		
Troba les claus	**Procés de desxifratge**	**Prova de seguretat**
Sí	Correcte	$m' = (-46, 10, -100)$
No	-	-
Sí	Correcte	$m' = (15, 78, -100)$
Sí	Correcte	$m' = (-46, 160, -100)$
No	-	-
Sí	Correcte	$m' = (-46, 226, -100)$
Sí	Correcte	$m' = (-46, 77, -100)$
Sí	Correcte	$m' = (-46, 78, 17)$
No	-	-
Sí	Correcte	$m' = (207, 78, -100)$

Taula 5.1: Resultats per a $k = 0.5$

Per a $k = 0.5$ veiem que sovint no es troben les claus, això és pel fet que la base aleatòria generada no es tracta d'una bona base, és a dir, és poc ortogonal. En els casos en què sí que es troben les claus, veiem que el grau de seguretat del criptosistema és molt baix, ja que disposant només de la base dolenta, l'Eve pot arribar a trobar un missatge molt semblant a l'original.

$k = 0.001$, $n = 3$, $m = (-46, 78, -100)$		
Troba les claus	**Procés de desxifratge**	**Prova de seguretat**
Sí	Correcte	$m' = (4985, 78, -95)$
Sí	Correcte	$m' = (43, -15, 28073)$
Sí	Incorrecte	$m' = (701, 81, 2380)$
Sí	Correcte	$m' = (-46, -40, -601)$
Sí	Incorrecte	$m' = (250, -1452, -102)$
Sí	Correcte	$m' = (-16, 78, 23716)$
Sí	Incorrecte	$m' = (-33463967, 75, -86344)$
Sí	Correcte	$m' = (11603, 78, -43)$
Sí	Correcte	$m' = (-120, 5887, -100)$
Sí	Correcte	$m' = (5, 71, 5986)$

Taula 5.2: Resultats per a $k = 0.001$

Per a $k = 0.001$ observem que en general es troben les claus, però sovint, a l'hora de realitzar el procés de desxifratge, l'Alice obté un missatge erroni. A més, l'Eve, fent ús de la clau pública segueix trobant un missatge semblant a l'original, fet que redueix el grau de seguretat del criptosistema.

$k = 0.00001$, $n = 3$, $m = (-46, 78, -100)$		
Troba les claus	**Procés de desxifratge**	**Prova de seguretat**
Sí	Correcte	$m' = (-133, 18648024, 50583)$
Sí	Correcte	$m' = (4387, 5108, 2866814)$
Sí	Correcte	$m' = (6547, -5669816, -100)$
Sí	Correcte	$m' = (-46, 18680, -1469565)$
Sí	Correcte	$m' = (-175, 72664, 82421)$
Sí	Correcte	$m' = (-46, 114382, -57266409)$
Sí	Incorrecte	$m' = (-46, -33542090, 87249)$
Sí	Incorrecte	$m' = (446766, 78, -247981111)$
Sí	Correcte	$m' = (-113, -10804, -43)$
Sí	Correcte	$m' = (55, 11228055, -16819)$

Taula 5.3: Resultats per a $k = 0.00001$

Finalment, per a $k = 0.00001$ la probabilitat de trobar les claus i que aquestes funcionin ja és molt més elevada en comparació amb el cas de $k = 0.5$. Encara que segueix existint casos en què la clau privada de l'Alice troba un missatge erroni, la resta de casos funciona correctament i, a més, el criptosistema és considerablement segur, ja que el missatge que es troba fent ús de la clau pública és molt diferent al missatge original.

Per tant, en reticles 3-dimensionals podríem concloure que amb un paràmetre k més petit que 0.00001 el criptosistema sembla que té un grau de seguretat prou elevat. És clar que no podem dir amb certesa que amb aquest valor de k el criptosistema serà segur per a tota base, ja que, tal i com hem vist a la Taula 5.3 encara existeixen casos en els quals el missatge obtingut a partir de la clau privada és erroni. Cal remarcar que aquest raonament és només aplicable per a reticles de dimensió 3. Per a dimensions més elevades caldrien fer més execucions i estudis dels resultats.

5.3.2 Algorisme LLL

Com hem esmentat anteriorment, l'algorisme LLL destaca per ser una gran amenaça per la seguretat del criptosistema GGH, ja que en dimensions baixes pot arribar a trencar el criptosistema. Tot i així, en dimensions elevades, l'algorisme deixa de ser eficient.
Dedicarem aquest apartat per a analitzar els resultats d'implementar l'algorisme LLL en diferents dimensions i experimentar amb la dimensió més gran a la que arriba l'algorisme amb eficiència.

$k = 0.00001$	
Dimensió del reticle	**Temps en trobar la base LLL reduïda**
3	0.07606983 s
5	0.3734219 s
7	1.1915330 s
9	4.1398570 s
11	31.8561470 s
13	49.5694169998 s
15	107.6636118888855 s
17	182.21712589263916 s
18	285.9577307701111 s

Taula 5.4: Temps d'execució de l'algorisme LLL en diferents dimensions

Veiem que a mesura que augmenta la dimensió del reticle, el temps d'execució també augmenta. Observem a la Taula 5.4 que en dimensió 18 l'algorisme ja triga més de 4 minuts en trobar la base LLL reduïda i finalitzar. Encara que aquestes execucions han estat realitzades des d'un ordinador portàtil i potser el codi no és el més eficient possible, aquestes dades ens donen una idea del que podria arribar a trigar a executar-se l'algorisme en dimensions superiors a 300. És per això que diem que l'algorisme LLL no és eficient en dimensions molt elevades i, per tant, no és capaç de trencar completament el criptosistema GGH.

Capítol 6

Conclusions

Durant el treball hem observat que amb el pas dels anys han anat sorgint diferents algorismes amb l'objectiu d'atacar la seguretat del criptosistema GGH. El més efectiu ha estat l'algorisme LLL que arriba a trencar el criptosistema fins a dimensions superior a 350. Encara així, l'algorisme LLL no ha sigut capaç de trencar-ho completament, ja que per dimensions molt elevades deixa de ser eficient. Malauradament, en dimensions molt elevades el criptosistema GGH també deixa de ser pràctic, ja que el tamany de les claus creixen en proporció a la dimensió. Encara així, aquest fet no treu mèrit i importància al criptosistema GGH, ja que va ser un dels criptosistemes que va introduir la criptografia basada en reticles. Darrerement, amb l'aparició del concepte de criptografia postquàntica s'ha vist que molts bons candidats per a criptosistemes resistents als atacs quàntics estan basats o inspirats en el criptosistema GGH i la criptografia basada en reticles.

Bibliografia

[1] Hoffstein, Jeffrey; Pipher, Jill; Silverman, Joseph H. *An introduction to mathematical cryptography,* segona edició, *Undergraduate Texts in Mathematics*. Springer, New York, 2014.

[2] Neukirch, Jürgen. *Algebraic number theory.* Translated from the 1992 German original and with a note by Norbert Schappacher. With a foreword by G. Harder. Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], 322. Springer-Verlag, Berlin, 1999. (pàg 25)

[3] Galbraith, Steven D. *Mathematics of public key cryptography*, Cambridge University Press, Cambridge, 2012.

[4] Nguyen, Phong Q.; Stehlé, Damien. *Low-dimensional lattice basis reduction revisited.* Algorithmic number theory, 338–357, Lecture Notes in Comput. Sci., 3076, Springer, Berlin, 2004.

[5] Lenstra, A.K.; Lenstra, H.W., Jr.; Lovász, L. *Factoring polynomials with rational coefficients.* Math. Ann. 261 (1982), no. 4, 515–534.

[6] Micciancio, D.; Goldwasser, S. *Complexity of Lattice Problems. A cryptographic perspective.* The Kluwer International Series in Engineering and Computer Science, 671. Kluwer Academic Publishers, Boston, MA, 2002.

[7] Curtis, C.W. *Linear algebra. An introductory approach.* Fourth edition. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1984.

[8] Phong Nguyen. *Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97*, 1999.

[9] Han, Daewan; Kim, Myung-Hwan; Yeom, Yongjin. *Cryptanalysis of the Paeng-Jung-Ha cryptosystem from PKC 2003.* Public key cryptography—PKC 2007, 107–117, Lecture Notes in Comput. Sci., 4450, Springer, Berlin, 2007

[10] Kisbye, N.P. *Sistemas de claves públicas y privadas.* Revista De Educación Matemática, 2021.

[11] Stallings, W. *Cryptography and Network Security.Principles and Practice.* Seventh edition, 1998.

[12] Micciancio, D.; Regev, O. *Lattice-based cryptography.* Post-quantum cryptography, 147–191, Springer, Berlin, 2009.

[13] Ajtai, M.; Dwork, C. *A public-key cryptosystem with worst-case/average-case equivalence.* STOC '97 (El Paso, TX), 284–293, ACM, New York, 1999.

[14] Goldreich, O.; Goldwasser, S.; Halevi, S. *Public-key cryptosystems from lattice reduction problems.* Advances in cryptology—CRYPTO '97 (Santa Barbara, CA, 1997), 112–131, Lecture Notes in Comput. Sci., 1294, Springer, Berlin, 1997.

[15] Hoffstein, J.; Pipher, J.; Silverman, J.H. *NTRU: a ring-based public key cryptosystem.* Algorithmic number theory (Portland, OR, 1998), 267–288, Lecture Notes in Comput. Sci., 1423, Springer, Berlin, 1998.

[16] Regev, O. *On lattices, learning with errors, random linear codes, and cryptography.* STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 84–93, ACM, New York, 2005.

[17] Ajtai, M. *Generating hard instances of lattice problems.* Quaderni di Matematica, 13:1–32, 2004. Preliminary version in STOC 1996.

[18] Zhaofei, Tian. *GGH Cryptosystem and Lattice Reduction Algorithms.* University of Peking, Beijing, 2011.

[19] Buchmann, J. *The Digital Signature Algorithm (DSA)*, 2001.

[20] Cordoba, D.; Méndez, M. *Criptografía Post Cuántica.* universidad de Mendoza.

[21] Peikert, C. *A Decade of Lattice Cryptography*, 2016.

Apèndix A

Criptosistema GGH

GGH

June 12, 2023

[ ]:
```
# CREACIÓ DE LES CLAUS

# 1. Creem una base bona

# Determinem la dimensió de la base i el paràmetre k:
n = 3
k = 0.000001

# Fixem un parametre d i escollim aleatòriament les coordenades dels vectors␣
 ↪entre -d i d:
d = 20
V =[]
for i in range (0,n):
    vi = []
    for j in range (0,n):
        vr = ZZ.random_element(-d,d+1)
        vi.append(vr)
    V.append(vi)
V = Matrix(V)

# 2. Comprovem que és una bona base

# Calculem el det(L) i comprovem que és diferent de 0:
det = V.det().abs()
print("det = " + str(det))
if det == 0:
    print("Els vectors generats no formen una base!")

else:
    # Calculem el coeficient de Hadamard:
    N = 1
    for i in range (0,n):
        norma = RR(V.row(i).norm())
        N = N*norma
    H = (det/N)^(1/n)

    #Comprovem que és proper a 1:
```

```
    if H > k:
        print("La base V és bona, ja que té un quocient de Hadamard = " +␣
↪str(H) + ".")
        print("La clau privada de l'Alice és la base:")
        print(V)

        # 3. Creem una base dolenta

        U = identity_matrix(n) # U = id
        W = U * V # W = U*V = V

        # Calculem el seu quocient de Hadamard:
        det = W.det().abs()
        N = 1
        for i in range (0,n):
            norma = RR(W.row(i).norm())
            N = N*norma
        H = RR((det/N)^(1/n))

        # Multipliquem per matrius elementals fins a trobar una base dolenta:
        while H > k:
            # Multipliquem per una matriu elemental:
            Z = identity_matrix(n)
            i = ZZ.random_element(0, n)
            j = ZZ.random_element(0, n)
            while j == i:
                j = ZZ.random_element(0, n)
            a = ZZ.random_element(-1000, 1000)
            Z.add_multiple_of_row(i, j, a)
            U = U*Z
            W = U*V
            # Calculem el seu quocient de Hadamard:
            det = W.det().abs()
            N = 1
            for i in range (0,n):
                norma = RR(W.row(i).norm())
                N = N*norma
            H = RR((det/N)^(1/n))
        detU = U.det()
        print("U = " + str(U))
        print("detU = " + str(detU))


        # La base resoltant serà una base dolenta ja que ha sortit del while.
        print("La base W és dolenta, ja que té un quocient de Hadamard = " +␣
↪str(H) + ".")
        print("La clau pública és la base:")
```

```
        print(W)

    else:
        print("La base és dolenta!")
```

[ ]:
```
# PROCÉS DE XIFRATGE

# 1. En Bob introdueix el missatge a xifrar
m = vector([-46,78,-100])

# 2. Tria un vector aleatori r petit
r =[]
delta = 2
for i in range (0,n):
    rr = ZZ.random_element(-delta, delta)
    r.append(rr)
r = vector(r)
print("r = " + str(r))

# 3. Calcula el text xifrat e
e = zero_vector(n)
e = m*W + r
print("El text xifrat del Bob és: " + str(e))
```

[ ]:
```
# PROCÉS DE DESXIFRATGE

# 1. L'Alice expressa el text xifrat e en funció de la base privada
Inv = V.inverse()
T = vector(e*Inv)
print("T = " + str(T))

# 2. L'Alice aplica l'algorisme de Babai sobre la seva clau privada
a = zero_vector(n)
v = zero_vector(n)
for i in range(0,n):
    a[i] = round(T[i]) # Arrodonim els coeficients a l'enter més pròxim
    print("ai = " + str(a[i]))
    v = v + a[i]*V.row(i) # El vector resultant és mW
print(v)

# 3. Ara multiplica per W^-1 i recupera m
msol = v*W.inverse()
print("m = " + str(msol))
```

[ ]:
```
# PROVA DE SEGURETAT

# 1. L'Eve expressa el text xifrat e en funció de la base pública
```

```
InvE = W.inverse()
TE = e*InvE
print("TE = " + str(TE))

# 2. L'Eve aplica l'algorisme de Babai sobre la clau pública
aE = zero_vector(n)
vE = zero_vector(n)
for i in range(0,n):
    aE[i] = round(TE[i])
    print("aEi = " + str(aE[i]))
    vE = vE + aE[i]*W.row(i)
print("v = " + str(vE))

# 3. Ara multiplica per W^-1
msol = vE*W.inverse()
print("m' = " + str(msol))
```

[ ]:
```
# COMPARACIÓ DE LES DUES CLAUS

error1 = e-v
error2 = e-vE
norm1 = RR(error1.norm())
norm2 = RR(error2.norm())
print("error1 = " + str(norm1))
print("error2 = " + str(norm2))
```

Apèndix B

Algorisme LLL

LLL

June 13, 2023

[ ]:
```
# ALGORISME LLL

MR = MatrixSpace(QQ,n,n)
v_LLL = copy(W)
k = 1

# Trobem la base ortogonal de Gram-Schmidt associada
v_GS = MR.matrix()
v_GS[0] = v_LLL.row(0)
for i in range (1,n):
    sumatori = zero_vector(QQ, n)
    for j in range (0,i):
        prod = v_LLL[i].dot_product(v_GS[j])
        norm = QQ(v_GS[j].norm()^2)
        mu = QQ(prod/norm)
        sumatori = sumatori + (mu*v_GS[j])
    v_GS[i] = v_LLL[i] - sumatori

v_GS[0] = v_LLL.row(0)

inici = time.time()
while k <= (n-1):
    for j in range (0,k):

        # Fem la reducció de mida
        prod = QQ(v_LLL[k].dot_product(v_GS[k-1-j]))
        norm = QQ((v_GS[k-1-j].norm())^2)
        mu = round(prod/norm)
        v_LLL[k] = v_LLL[k] - (mu*v_LLL[k-1-j])

        # Actualitzem la base de Gram-Schmidt
        v_GS[0] = v_LLL.row(0)
        for i in range (1,n):
            sumatori = zero_vector(QQ, n)
            for j in range (0,i):
                prod = v_LLL[i].dot_product(v_GS[j])
                norm = QQ(v_GS[j].norm()^2)
```

```
                mu = QQ(prod/norm)
                sumatori = sumatori + (mu*v_GS[j])
            v_GS[i] = v_LLL[i] - sumatori

    # Comprovem que se satisfà la condició de Lovász
    normk = QQ(v_GS[k].norm()^2)
    prodk = QQ(v_LLL[k].dot_product(v_GS[k-1]))
    normkk = QQ(v_GS[k-1].norm()^2)
    muk = QQ(prodk/normkk)
    if normk >= (((3/4) - (muk^2))*normkk):
        k = k+1

    else:
        w = copy(v_LLL[k-1])
        v_LLL[k-1] = copy(v_LLL[k])
        v_LLL[k] = copy(w)
        k = max(k-1, 1)

        # Actualitzem la base de Gram-Schmidt
        v_GS[0] = v_LLL.row(0)
        for i in range (1,n):
            sumatori = zero_vector(QQ, n)
            for j in range (0,i):
                prod = v_LLL[i].dot_product(v_GS[j])
                norm = QQ(v_GS[j].norm()^2)
                mu = QQ(prod/norm)
                sumatori = sumatori + mu*v_GS[j]
            v_GS[i] = v_LLL[i] - sumatori

final = time.time()
print(final - inici)

print("v_LLL = ")
print(v_LLL)
```

[ ]:
```
# TRENCAMENT AMB LLL

# 1. L'Eve calcula la base LLL reduïda i comprova que és una bona base.

# Calculem el seu quocient de Hadamard
det = v_LLL.det().abs()
N = 1
for i in range (0,n):
    norma = v_LLL.row(i).norm()
    N = N*norma
H = RR((det/N)^(1/n))
print("Quocient de Hadamard de la base LLL reduïda: " + str(H))
```

```
# 2. L'Eve expressa el text xifrat e en funció de la base LLL reduïda.
InvE = v_LLL.inverse()
TE = e*InvE
print(TE)

# 3. L'Eve aplica l'algorisme de Babai sobre la base LLL reduïda.
aE = zero_vector(n)
vE = zero_vector(n)
for i in range(0,n):
    aE[i] = round(TE[i])
    print(aE[i])
    vE = vE + aE[i]*v_LLL.row(i)
print(vE)

# Ara multiplica per W^-1 i recupera el missatge m
msol = vE*W.inverse()
print(msol)
```

[ ]: