Introduction to LLL
“Cryptography”

Di Santi Giovanni

May 27, 2021

Contents

Chapter 1

Linear Algebra Background

1.1 Vector Spaces

Definition 1.1.1 ***Vector space.***

A `vector space` V is a subset of $\mathbb{R}^m$ which is closed under finite vector addition and scalar multiplication, with the property that

$$a_1 v_1 + a_2 v_2 \in V \text{ for all } v_1, v_2 \in V \text{ and all } a_1, a_2 \in \mathbb{R}$$

Definition 1.1.2 ***Linear Combinations***

Let $v_1, v_2, \ldots, v_k \in V$. A `linear combination` of $v_1, v_2, \ldots, v_k \in V$ is any vector of the form

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k \text{ with } \alpha_1, \ldots, \alpha_k \in \mathbb{R}$$

Definition 1.1.3 ***Lineaer Independece***

A set of vectors $v_1, v_2, \ldots, v_k \in V$ is `linearly independent` if the the only way to get

$$a_1 v_1 + a_2 v_2 + \cdots + a_k v_k = 0$$

is to have $a_1 = a_2 = \cdots = a_k = 0$.

Definition 1.1.4 ***Bases***

Taken a set of linearly independent vectors $b = (v_1, \ldots, v_n) \in V$ we say that b is a `basis` of V if $\forall w \in V$ we can write

$$w = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$$

Definition 1.1.5 ***Vector's length***

The vector's length or `Euclidean norm` of $v = (x_1, x_2, \ldots, x_m)$ is

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

Definition 1.1.6 ***Dot Product***

Let $v, w \in V \subset \mathbb{R}^m$ and $v = (x_1, x_2, \ldots, x_m), w = (y_1, y_2, \ldots, y_m)$, the `dot product` of v and m is

$$v \cdot m = x_1 y_1 + x_2 y_2 + \cdots + x_m y_m$$
or
$$v \cdot m = \|v\| \|w\| \cos \theta$$

where θ is the angle between v and w if we place the starting points of the vectors at the origin O.

Geometrically speaking $v \cdot m$ is the length of w projected to v multiplied by the length of v as shown in 1.1

Definition 1.1.7 ***Ortoghonal Basis***

An `ortoghonal basis` for a vector space V is a basis $v_1, \ldots, v_m$ with the property that

$$v_i \cdot v_j = 0 \text{ for all } i \neq j$$

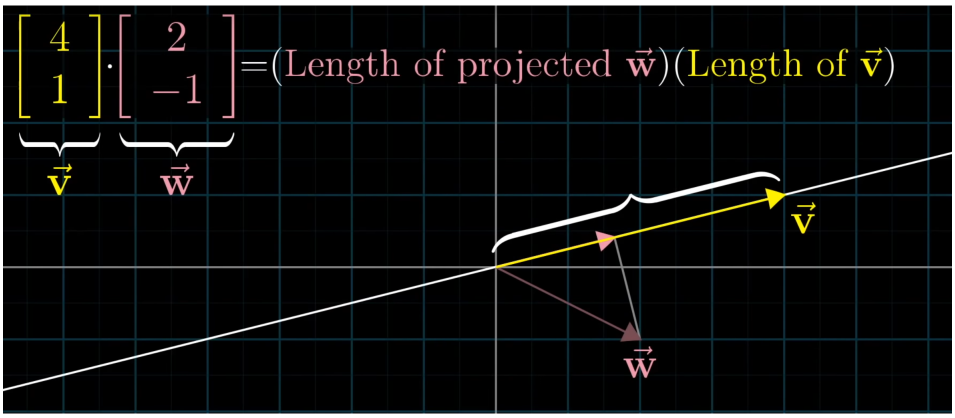

Figure 1.1: Dot Product By 3Blue1Brown

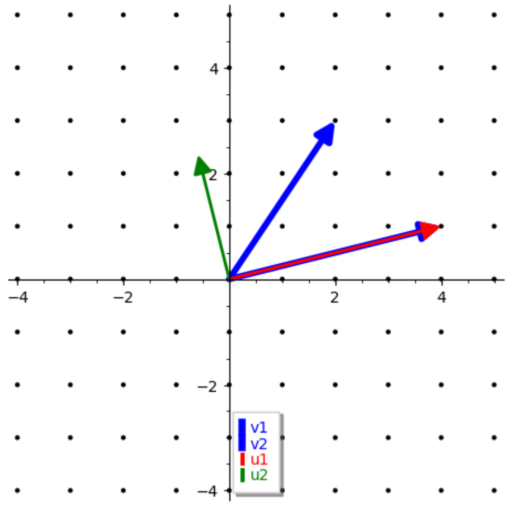

Figure 1.2: Gram Schmidt orthogonalization

If $\|v_i\| = 1$ for all i then the basis is `orthonormal`.

Let $b = (v_1, \ldots, v_n)$, be a basis for a vector space $V \subset \mathbb{R}^m$. There is an algorithm to create an orthogonal basis $b^* = (v_1^*, \ldots, v_n^*)$. The two bases have the property that $\mathrm{Span}\{v_1, \ldots, v_i\} = \mathrm{Span}\{v_1^*, \ldots, v_i^*\}$ for all $i = 1, 2, \ldots, n$

If we take $v_1 = (4, 1), v_2 = (2, 3)$ as basis and apply gram schmidt we obtain $u_1 = v_1 = (4, 1), u_2 = (-10/17, 40/17)$ as shown in 1.2

1.2 Lattices

Definition 1.2.1 *Lattice*

Let $v_1, \ldots, v_n \in \mathbb{R}^m, m \geq n$ be linearly independent vectors. A `Lattice` L spanned by $\{v_1, \ldots, n_n\}$ is the set of all integer linear combinations of $v_1, \ldots, v_n$.

$$L = \left\{ \sum_{i=1}^{n} a_i v_i, a_i \in \mathbb{Z} \right\}$$

If v_i for every $i = 1, \ldots n$ has integer coordinates then the lattice is called `Integral Lattice`.

On the figure 1.3 we show a lattice L with bases $v = (3, 1)$ and $w = (-1, 1)$, and on 1.4 the same lattice L with a different basis.

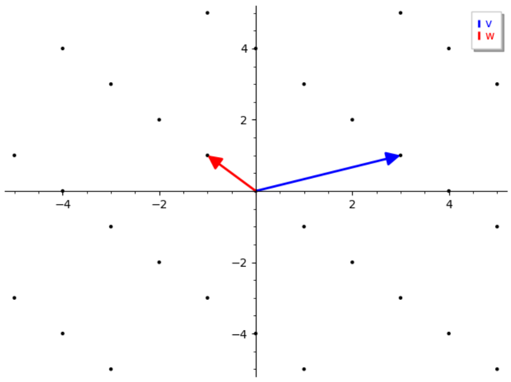

Figure 1.3: Lattice L spanned by v, w

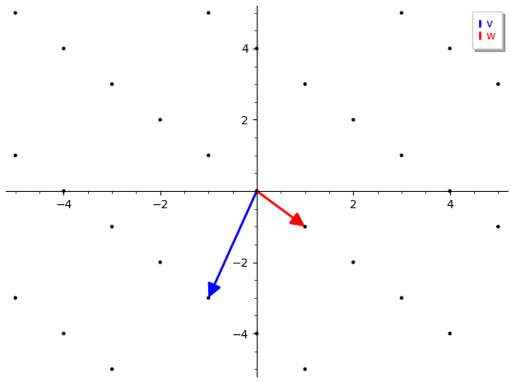

Figure 1.4: Lattice L spanned by v', w'

1.3 Problems

1.3.1 SVP

The Shortest Vector Problem (`SVP`): Find a nonzero vector $v \in L$ that minimez the Euclidean norm $\|v\|$.

Gauss Reduction

Gauss's developed an algorithm to find an optimal basis for a two-dimensional lattice given an arbitrary basis. The output of the algorithm gives the shortest nonzero vector in L and in this way solves the `SVP`.

If we take for example $v_1 = (10, 4), v_2 = (7, 5)$ and apply the gauss reduction algorithm we obtain $w_1 = (3, -1), w_2 = (4, 6)$ 1.5. w_1 is the shortest nonzero vector in the lattice L spanned by v_1, v_2.

However the bigger the dimension of the lattice, the harder is the problem and there isn't a polynomial algorithm to find such vector.

1.3.2 CVP

The Closest Vector Problem (`CVP`): Given a vector $w \in \mathbb{R}^m$ that is not in L, find a vector $v \in L$ that is closest to w, in other words find a vector $v \in L$ that minimizes the Euclidean norm $\|w - v\|$.

Example in 1.6

TODO: CVP and SVP are related.

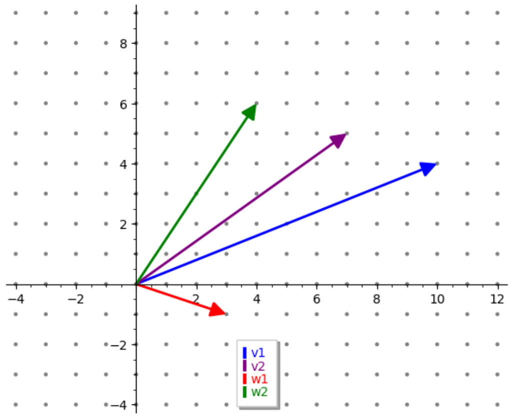

Figure 1.5: Gauss reduction

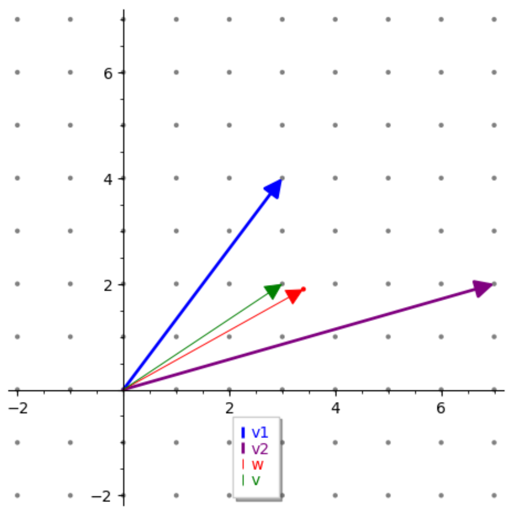

Figure 1.6: CVP

Chapter 2

LLL

2.1 Introduction

The **Lenstra-Lenstra-Lovász** *LLL* or L^3 is a polynomial time algorithm to find a "shorter" basis.

Theorem 2.1.1 ***LLL***

Let $L \in \mathbb{Z}^n$ be a lattice spanned by $B = \{v_1, \ldots, v_n\}$. The LLL algorithm outputs a reduced lattice basis $\{w_1, \ldots, w_n\}$ with

$$\|w_i\| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} det(L)^{\frac{1}{n-i+1}} \text{ for } i = 1, \ldots, n$$

in time polynomial in n and in the bit-size of the entries of the basis matrix B.

Basically the first vector of the new basis will be as short as possible, and the other will have increasing lengths. The new vectors will be as orthogonal as possible to one another, i.e., the dot product $w_i \cdot w_j$ will be close to zero.

Example

For example we can take the following basis (the rows are the vector) that span a lattice L.

$$L = \begin{pmatrix} 4 & 9 & 10 \\ 2 & 1 & 30 \\ 3 & 7 & 9 \end{pmatrix}$$

Applying the LLL algorithm we obtain

$$LLL(L) = \begin{pmatrix} -1 & -2 & -1 \\ 3 & -2 & 1 \\ -1 & -1 & 5 \end{pmatrix}$$

Where the first row is the shortest vector in the lattice L, and so solves the **SVP** problem. For higher dimensions however the LLL algorithm outputs only an approximation for the **SVP** problem.

2.2 Algorithm

TODO: Write algorithm and explain some steps

2.3 Applications

There are many applications of LLL

1. Factoring polynomials over the integers. For example, given $x^2 - 1$ factor it into $x + 1$ and $x - 1$.
2. Integer Programming. This is a well-known **NP**-complete problem. Using LLL, one can obtain a polynomial time solution to integer programming with a fixed number of variables.
3. Approximation to the **CVP** or **SVP**, as well as other lattice problems.
4. Application in cryptanalysis.

Chapter 3

Cryptanlysis

3.1 RSA introduction

RSA is one of the earliest and most used asymmetric cryptosystem. The usual step to generate a public/private key for **RSA** is the following

1. Fix $e = 65537$ or $e = 3$ (public).
2. Find two primes p, q such that $p - 1$ and $q - 1$ are relatively prime to e, i.e. $\gcd(e, p - 1) = 1$ and $\gcd(e, q - 1) = 1$.
3. Compute $N = p * q$ and $\phi(n) = (p - 1) * (q - 1)$
4. Calculate d (private) as the multiplicative inverse of e modulo $\phi(n)$.
5. (N, e) is the public key, (N, d) is the private key.

To encrypt a message m with **textbook RSA**

$$c = m^e \mod N$$

To decrypt a ciphertext c

$$m = c^d \mod N$$

3.2 Lattices against RSA

It's easy to find the roots of a univariate polynomial over the integers. Finding the roots of **modular** polynomial is hard, example:

$$f(x) \equiv 0 \mod N$$

Suppose N is an **RSA** modulus and we don't know the factorization of it. Let's have an univariate integer polynomial $f(x)$ with degree n

$$f(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 n + a_0$$

Coppersmith showed how we can recover the value x_0 such that $f(x_0) \equiv 0 \mod N$, with $x_0 < N^{\frac{1}{n}}$ in polynomial time using the following theorem

Theorem 3.2.1 ***Howgrave-Graham***

Let $g(x)$ be an univariate polynomial with n monomials and m be a positive integer. If we have some restraint X and the following equations hold

$$g(x_0) \equiv 0 \mod N^m, |x_0| \leq X \tag{3.1}$$
$$\|g(xX)\| < \frac{N}{\sqrt{n}} \tag{3.2}$$

Then $g(x_0) = 0$ holds over the integers.

This theorem states that is possible to compute the root of $f(x) \mod N$ if we can find a polynomial $g(x)$ that share the same root but modulo N^m. If 3.1 and 3.2 hold then we can simply compute the root of $g(x)$ over the integers to have the same root x_0 such that $f(x_0) \equiv 0 \mod N$.

Howgrave-Graham's idea is to find this polynomial g by combining polynomials $g_{i,j}$ who also have x_0 as roots modulo N^m.

The **LLL** algorithm is fundamental because:

- It only does **integer linear operations** on the basis vectors. In this way even if the basis is different it's only a linear combination of vector that still have x_0 as root modulo N^m.
- If we craft the lattice properly, the norm of shortest vector on the reduced basis will satisfy 3.2.

We can easily create polynomials p_i ($g_{i,j}$ and h_i) sharing the same root x_0 over N^m of f where δ is the degree of f:

$$g_{i,j}(x) = x^j \cdot N^i \cdot f^{m-i}(x) \text{ for } i = 0, \ldots, m-1, \; j = 0, \ldots, \delta - 1 \tag{3.3}$$
$$h_i(x) = x^i \cdot f^m(x) \text{ for } i = 0, \ldots, t-1 \tag{3.4}$$

3.2.1 Example

We have a 100-bit **RSA** modulus

$$N = 0xf046522fb555a90bdc558fc93 \text{ and } e = 3.$$

Before the encryption the message m is padded as

$$z = pad||m = 0x74686973206b65793a||m$$

where $||$ is the concatenation. The padding is the ascii encoding of “this key:”

The ciphertext is

$$c = z^e \mod N = 0x5b603cda4b72100c6f25954fc$$

Suppose that we don't know the factorization of N and we would like to know the message m. However we know the padding and that the length of $m < 2^{16}$.

Let's define

$$a = 0x74686973206b65793a0000.$$

which is the known padding string that got encrypted.

Thus we have that $c = (a + m)^3 \mod N$, for an unkown small m. We can define $f(x) = (a + x)^3 - c$, and so we setup the problem to find a small root m such that $f(m) \equiv 0 \mod N$

$$f(x) = x^3 + 0x15d393c596142306bae0000x^2 + 0x1b53c5e184a49b39f9ad9eedbx \\ +0x486a5d936fb568185c8ff0506$$

Lattice contruction. Let the coefficients of f be $f(x) = x^3 + f_2x^2 + f_1x + f_0$ and $X = 2^{16}$ be the upper bound of the size of the root m. We can construct the matrix

$$B = \begin{pmatrix} X^3 & f_2X^2 & f_1X & f_0 \\ 0 & NX^2 & 0 & 0 \\ 0 & 0 & NX & 0 \\ 0 & 0 & 0 & N \end{pmatrix}$$

The rows of the matrix correspond to the coefficient vectors of the polynomials $f(x), Nx^2, Nx$ and N, furthermore we know that each polynomials will be 0 modulo N if evaluated at $x = m$. We didn't fully applied Howgrave-Graham because we only used $m = 1$ (the N^m parameter not the message). With this lattice construction every vector is of the form $v = (v_1X^3, v_2X^2, v_1X, v_0)$, because any integer linear combination of the vector of the lattice will keep the bound X^i for $i = 0, .., \dim(B) - 1$.

Apply LLL. We then apply LLL to find the shortest vector of the reduced basis:

$$v = (0x90843131bc53X^3 + 0x2736f60b1c7ba3294X^2,$$
$$-0x1bec331b20625341b6d73X, 0x47336b98335c143ac912ec9e)$$

We can construct the polynomial g using the coefficients of v

$$g(x) = 0x90843131bc53x^3 + 0x2736f60b1c7ba3294x^2$$
$$-0x1bec331b20625341b6d73x + 0x47336b98335c143ac912ec9e$$

We know that

$$g(x_0) \equiv 0 \mod N, |x_0| \leq X$$

What we need to prove is that

$$\|g(xX)\| \leq \frac{N}{\sqrt{n+1}}$$

In this example, $\det B = X^6N^3$, and LLL will find a short vector with $\|v\| \leq 1.02^n(\det B)^{\frac{1}{dimB}}$. If we ignore the 1.02^n factor, then we need to satisfy that

$$g(m) \leq \|v\| \leq (\det B)^{\frac{1}{4}} < X$$

We have $(\det B)^{\frac{1}{4}} = (X^6N^3)^{\frac{1}{4}} < N$, if we solves for X this will be satisfied when $X < N^{\frac{1}{6}}$. In this case we have $N =$ 100-bit number and $X = 2^{16}$, so $N^{\frac{1}{6}} > X$ and we have verified the correctness.

If we compute the root of $g(x)$ over the integers we obtain $m = 0x6162$ which is the correct result.

This specific lattice works to find roots up to size $N^{\frac{1}{6}}$, so the same construction will work if we want to find

- ~170 unkown bits of message from an RSA 1024-bit modulus
- ~341 unkown bits of message from an RSA 2048-bit modulus
- ~683 unkown bits of message from an RSA 4096-bit modulus

If you want to find bigger root you need to create a bigger lattice with more polynomials generated with 3.3, this method is better described in TODO, but the principles are the same.

3.3 Key recovery ECDSA

End of Paper

gg^2

Bibliography