Bases de Groebner y aplicaciones a sistemas criptográficos

Almudena Agudo

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Director del trabajo: José Ignacio Cogolludo y Jorge Martín Morales

28 de junio del 2017

Prólogo

Ante la vertiginosa amenaza a nuestros sistemas criptográficos modernos debido a los avances tecnológicos en el área de computación y la inminente aparición de los conocidos como ordenadores cuánticos, se hace necesaria la búsqueda de nuevos sistemas criptográficos. Es en este contexto donde basaremos nuestro trabajo.

Algunas de estas propuestas utilizan anillos de polinomios multivariantes en lugar de la aritmética modular. En este trabajo nos centraremos en la descripción de algunas de dichas propuestas, así como en un modesto estudio de su robustez basado en ataques propuestos en la literatura utilizando el servidor de Sage de la Universidad.

El paquete de cálculo Sage ha sido frecuentemente utilizado a lo largo del grado, lo que aportó una familiaridad respecto a su uso, aunque también ha sido necesaria un nuevo aprendizaje de programación en Sage. Otras asignaturas básicas para entender este trabajo son Estructuras Algebráicas y Álgebra aplicada y computacional. La primera nos ha ayudado en el fundamento teórico del trabajo en términos de aritmética modular, anillos de polinomios, cuerpos de números y extensiones de cuerpos finitos. La segunda nos ha aportado la visión general de la criptografía moderna, sus objetivos, sus métodos y sus aplicaciones. Hemos necesitado ampliar conocimientos en bases de Groebner, aplicaciones polinómicas, algoritmos de la división en varias variables y un acercamiento a la aplicación de las bases de Groebner a diversos problemas matemáticos, entre ellos cabe destacar la coloración de grafos o el problema del grafo perfecto, por último añadir las propuestas de ataques a sistemas criptográficos de tipo Polly-Cracker.

En los ejemplos de ataque a criptosistemas de Polly-Cracker estudiados comprobamos que la dificultad de éxito crece rápidamente al aumentar el número de variables involucradas, como es de esperar. De hecho, la idea comúnmente aceptada entre los especialistas es que o bien este método es una alternativa a RSA o bien los ataques por medio de bases de Groebner no son eficaces.
El problema de encontrar ataques efectivos continúa abierto antes de aceptar que estos sistemas sean la alternativa adecuada a RSA.

El trabajo está distribuido de la siguiente forma, en el Capitulo 1 comenzaremos con un repaso a toda la historia de la criptografía, hablando de los tipos de criptografía existentes, para lo que nos hemos basado en [4]. Seguiremos con el método RSA, fundamental en la encriptación computacional, dicho algoritmo está basado en una aritmética modular que se apoyará en el Pequeño Teorema de Fermat y en la función de Euler, para lo cual hemos seguido [3].

A la llegada de la nueva era de ordenadores cuánticos el método RSA dejará de sernos útil puesto que romper este tipo de criptosistemas será poco costoso de tal forma vamos a introducir nuevos métodos para los cuales necesitaremos una fundamentación teórica de las Bases de Groebner, que estará explicada a lo largo del Capítulo 2. Nos hemos basado en el texto [1]. Trabajaremos con polinomios en varias variables con coeficientes en un cuerpo finito y hablaremos de la manera de ordenar los monomios en dicho anillo y de su división para poder llegar a ejecutar el algoritmo de Buchberger, que se usará para la creación de bases de Groebner.

En el Capítulo 3, explicaremos algunas de las alternativas propuestas al método RSA, como son Polly-Cracker y el Inverso polinómico. Dichos criptosistemas son susceptibles de ataques por medio de bases de Groebner, como se detalla en [2]. Por último, en el Capítulo 4 haremos un estudio cuantitativo de los tiempos de rotura del criptosistema de Polly-Cracker. Para ello hemos elaborado un código implementado en Sage.

Summary

Cryptography is the science based on the art of writing with an enigmatic method or with open methods and public keys, to hide a message so that only the person who has the appropriate secret key can read it, and the others cannot.

This work is divided in 4 chapters. An outline of each one of them will be given as a short description of each one.

Chapter 1: History of cryptography

The science of cryptography has been evolving throughout history. The origin of this science comes from the ancient Egypt due to the use the hieroglyphs, and that is considered to be the first way to encode messages, then that science was developing until reaching the top during World War II, with Alan Turing, a famous mathematician who invented the Enigma machine.

Then, the digital time arrived and with that computers too, in this moment a type of cryptographic digital system that is used to sign and encode digitally was developed, this method is called RSA, that uses a public key and is based on the breakdown of prime numbers. Hard process to current computers and here was the security of this method and here we obtain the security of this process but when quantum computers will be constructed this type of process will be left out-dated due to the breakdown time will be reduce and we need to create another methods.

There exists three types of cryptography methods:

- Cryptographyc symmetric system, we use only one key to code and encode.
- Cryptographyc asymmetric system, we use a public key and a private key.
- Cryptographyc mixed system, that is a mix of symmetric and asymmetric cryptosystems.

Chapter 2: Groebner's Bases

For the execution of new methods we need to introduce Groebner's Bases, a particular kind of generating set of an ideal in a polynomial ring $\mathbb{K}[x_1,..,x_n]$ over the field $\mathbb{K}$.

So the way to obtain a Groebner Basis comes from Buchberger's Algorithm, and to compute that, we will need to define the division algorithm in many variables, something that is difficult because first of all, we need to decide a monomial order to know how to start to divide.

The disadvantage of this algorithm is that sometimes we introduce redundant terms that we need to delete them, and for that we are going to learn how to do it when we delete the redundant terms we obtain a minimal or reduced Groebner Bases.

Chapter 3: New cryptographic methods

In this chapter we will talk about new algorithms because our RSA method begins to be threatened, due to the quantum computers arrive and with them the breakdown time is shorter than the current computers.

To describe them we use a polynomial ring $\mathbb{K}[x_1,..,x_n]$ over a field $\mathbb{K}$ and we are able to use, whenever we want to breakdown the algorithm to obtain the original message, Groebner's bases. These methods are called Polly-Cracker cryptosystem and the inverse of a polynomial map.

Chapter 4: Examples of breakdown time of Polly-Cracker cryptosystems

In this chapter we want to proof the strength of Polly-Cracker cryptosystems; for that we have done a analysis in Sage finding the breakdown time. The conclusion is that either the Groebner bases are not the best solution to obtain the original message or the Polly-Cracker cryptosystems are really difficult to break.

Due to the cryptography progress we need to adapt to the current problems and obtain solutions for that we are going to study Polly Cracker or Inverse Polynomial and we are trying to see if the breakdown time is hard when we want to find some solution to a Groebner basis.

Índice general

Capítulo 1

Historia de la criptografía

La criptografía es el arte de escribir con clave secreta o de un modo enigmático con el fin de cifrar o de codificar mensajes, evitando que su contenido pueda ser leído por un tercero no autorizado.
Su origen etimológico proviene de las palabras griegas *kryptós* que significa oculto, *grafein* cuyo significado quiere decir escribir y el sufijo *-ía* que tiene la función de ser utilizado para crear sustantivos abstractos. Si hablamos de la rotura de códigos del método criptográfico hablamos del hallazgo de descifrar el algoritmo.

Este fenómeno se remonta a hace más de 4.500 años situándonos en el antiguo Egipto donde da comienzo la criptografía por medio de la utilización de los jeroglíficos, se cree que estos no llegaron a ser una forma de comunicación secreta sino simplemente una búsqueda del despertar de la curiosidad, en este tipo de encriptación clásica se hace destacar el código César, proveniente de los romanos, el cifrado por transposición que es de origen griego o el código hebreo que es el llamado cifrado por sustitución.

Durante el medievo se desarrolló lo que actualmente llamamos criptografía medieval donde se profundizó en esta técnica por medio del descifrado por frecuencias que consiguió romper el método de los cifrados por sustitución, en esta época se creó un cifrado polialfabético, es decir un tipo de cifrado por sustitución donde cada carácter no se sustituye siempre por el mismo carácter.

Hasta la Segunda Guerra Mundial se desarrollaron nuevos métodos como el de Cuatro Cuadrados, el Doble Cuadrado y el ADFGVX entre otros muchos existentes. Aunque el mayor despunte de la criptografía se sitúa a partir de la Segunda Guerra Mundial donde mencionaremos al pionero Alan Turing ya que gracias a él se pudieron descifrar los códigos alemanes a partir de la creación de la conocida máquina Enigma. Posteriormente la criptografía avanzó vertiginosamente llegando al punto en que los avances de la rotura de códigos quedaron a manos del gobierno, los cuales se llegaron a declarar alto secreto.

A mitad del siglo XX gracias a los avances que se estaban produciendo en las matemáticas, a la llegada de los ordenadores y a una necesidad de ocultación de datos se da paso a la criptografía moderna y a la era digital donde destacaremos al estadounidense Claude Shannon recordado como “el padre de la información".

En este nuevo ciclo podremos hablar de la criptografía de clave pública o también llamada asimétrica, las claves que usa este método se denominan clave pública y clave privada, ambas posen unas características matemáticas especiales, es decir una serie de algoritmos complejos que dificultan la rotura del método. Siempre se generaran dos claves a la vez de manera que si dos claves públicas son distintas quiere decir que sus claves privadas también lo son.

Estos algoritmos están basados en las denominadas funciones hash, llamadas funciones de un único

sentido, es decir que a partir de una clave privada tenemos la imperiosa necesidad de generar una clave pública pero si nosotros poseemos la clave pública es realmente arduo concluir cual es la privada.

1.1. Tipos de criptografía

A lo largo de la historia como hemos visto, se han ido creando diversos tipos de criptografía, cada vez más evolucionados para evitar la rotura de los métodos y que nadie pudiera interceptar los mensajes enviados, así mientras unos algoritmos usan una misma clave tanto para cifrar como para descifrar, otros tienen una clave pública y otra privada y otros métodos llegan a ser incluso más complejos. Veamos los tres tipos de criptografía que se han ido desarrollando con el paso del tiempo.

- Criptografía simétrica
- Criptografía asimétrica
- Criptografía híbrida o mixta

Definición. La *criptografía simétrica* o también llamada *criptografía de clave secreta* es un método criptográfico en el cual se utiliza la misma clave tanto para cifrar y descifrar mensajes. Las dos partes que se quieren comunicar han de acordar de antemano qué clave que van a usar.

Como ejemplo principal de este tipo de criptografía se encuentra la máquina Enigma, aunque existen otros muchos como pueden ser: DES, 3DES, AES...

Definición. La *criptografía asimétrica* es el método criptográfico que usa un par de claves para la transmisión de mensajes: una servirá para encriptar, conocida como la *clave pública* que se encontrará al alcance de todo el mundo y otra, para desencriptar, esta será llamada *clave privada* que solo la posee la persona que ha de descifrar los mensajes para poder evitar que sean leídos por un tercero. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

El principal algoritmo de criptografía asimétrica es el método RSA, que describiremos posteriormente.

En la práctica, se utiliza un sistema que es combinación de los anteriores. Este permite alcanzar una seguridad alta (tal vez algo menor que el sistema asimétrico) con unos costes computacionales mucho menores. Este es el método criptográfico híbrido o mixto.

Definición. La *criptografía híbrida* es un método criptográfico que usa tanto un cifrado simétrico como un asimétrico. Cuando deseamos mandar un mensaje, este mensaje lo cifraremos con una clave simétrica, conocida en este proceso como la *clave de sesión*. Posteriormente tomaremos la clave pública del receptor del mensaje y lo cifraremos. Cuando el receptor reciba el mensaje, primero descifrará, con su clave privada, la clave de sesión. Una vez posee esta ya puede acceder al contenido del mensaje.

1.2. Descripción del método RSA

En 1977 se crea un método de clave pública llamado RSA, es un método criptográfico asimétrico más utilizado hoy en día cuyo nombre viene dado por las iniciales de sus creadores: Rivest, Shamir y Adleman. Este algoritmo es el primero que se usará tanto para cifrar como para firmar digitalmente. Se apoya en el manejo de dos claves; una pública, que se encuentra al alcance de todo el mundo y una

privada que la posee únicamente la persona que ha de descifrar el mensaje.

El esquema se basa en la existencia de un emisor de un mensaje, tradicionalmente llamada Alicia, un receptor que sería Bob y entre ellos una tercera persona, Eva, que quiere descifrar este mensaje.

El cometido es que Alicia envie un mensaje a Bob sin que Eva sea capaz de enterarse del contenido, de tal forma que Bob da a conocer su clave pública, aquella con la que todo el mundo puede cifrar los mensajes que desee enviarle, luego Alicia tomando esta clave y aplicándosela al mensaje junto con su propia clave privada hace que quede dicho mensaje encriptado y listo para enviárselo a Bob, este con la clave pública de Alicia desencriptaría la primera fase del mensaje pero todavía le quedaría otra etapa más para conseguir descifrar completamente el mesaje y para ello usará su clave privada, que solamente él conocerá, así podrá obtener el mensaje y saber qué quería decirle Alicia. Mientras ocurre todo este proceso Eva ve que se envía un mensaje y trata de interceptarlo sin conocer la clave privada de Bob, esto implica que Eve no logrará saber qué ha escrito Alicia porque existe una gran dificultad a la hora de llegar a comprender el mensaje si no posees la clave privada.

Traduciremos este esquema al método RSA, para poder comprender el algoritmo comenzaremos recordando la función de Euler.

Definición. Para cada $n \geq 1$, se define la función φ como:

$$\varphi(n) = \#\{k \in \mathbb{N} \mid k \leqslant n,\ mcd(k,n) = 1\}$$

es decir, la cantidad de números enteros positivos menores que n que son primos con n.

Tengamos en cuenta las propiedades de la función de Euler:

Propiedades 1.1.

- *n es primo si y solo si* $\varphi(n) = n-1$
- $\varphi(n^k) = (n-1)n^{k-1}$ *si n es primo y k es un número natural.*
- $\varphi(nm) = \varphi(n)\varphi(m)$ *si n y m son primos entre sí.*

Posteriormente, para llegar a la comprensión del método RSA, veamos qué nos dice el pequeño teorema de Fermat.

Teorema 1.1. *Si* $mcd(a,p) = 1$ *y p es primo, entonces* $a^{p-1} = 1 \mod p$.

Antes de empezar la demostración, recordemos que la aritmética modular puede ser construida matemáticamente mediante una relación de congruencia entre enteros, que es compatible con las operaciones de suma y multiplicación en el anillo de enteros.

Así pues, a y b se encuentran en la misma clase de congruencia $\mod n$, que escribiremos $a = b \mod n$ si ambos dejan el mismo resto cuando los dividimos entre n, o, equivalentemente, si $a-b$ es un múltiplo de n.

Procedamos con la demostración:

Demostración. Sea $\mathbb{Z}_p$ el anillo de los enteros módulo un número primo p. Esto nos asegura que $\mathbb{Z}_p$ es de hecho un cuerpo, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ que sabemos que es un grupo multiplicativo cíclico, es decir, que está generado por un solo elemento. Por lo tanto si $mcd(a,p) = 1$, entonces $a \in \mathbb{Z}_p^*$, y por lo tanto $a^{p-1} = 1 \mod p$. □

Veamos una variante del teorema que acabamos de enunciar utilizando la función de Euler.

Teorema 1.2. *Sea n libre de cuadrados, entonces* $a^{k\varphi(n)+1} \equiv a \mod n$.

El esquema general de transmisión del mensaje en RSA consiste en la existencia de dos aplicaciones $E, D : \mathbb{N} \to \mathbb{N}$ tales que E es una aplicación inyectiva, conocida como *función Hash* de modo que $DE(n) = n$. La robustez de este sistema se basa en la dificultad para obtener D a partir de E. El procedimiento consiste en, teniendo un mensaje m a enviar, entonces E se define a partir de la clave pública y D a partir de la clave privada.

Veamos su aplicación concreta:

1. Tomemos p y q dos números primos distintos y definamos $n = pq$. Recordemos que por propiedades 1.1 se cumple que $\varphi(n) = (p-1)(q-1)$.
2. Elijamos $e \in \mathbb{N}$ primo con $\varphi(n)$.
3. Calculemos un número entero d que sea el inverso de e módulo $\varphi(n)$, es decir, que $ed = 1 \mod \varphi(n)$. Esto es computacionalmente sencillo utilizando el algoritmo extendido de Euclides.
4. La aplicación de encripación E se define de la siguiente manera $E(m) = m^e \mod n$.
5. Cuando el receptor recibe el mensaje encriptado c puede utilizar la aplicación $D(c) = c^d \mod n$ para recuperar el mensaje original m. Comprobemos que este es el caso, es decir, que $D(E(m)) = m$. Efectivamente, dado que $ed = 1 \mod \varphi(n)$, podemos decir que $ed = 1 + k\varphi(n)$, entonces
$$D(m^e) = (m^e)^d = m^{1+k\varphi(n)} = m \mod n$$
la última igualdad se tiene por el teorema 1.2.

En términos práctico, si la longitud del mensaje fuera demasiado larga, este se divide en bloques y cada bloque será representado por un número menor que n. Por último notar que (e,n) juegan el papel de la clave pública mientras que d será la clave privada. La gran dificultad para obtener d a partir de la clave pública e y n estriba en el hecho de que se necesita el conocimiento de $\varphi(n)$ para el cual se requiere la factorización de n en sus factores primos, problema computacionalmente muy complejo para enteros suficientemente grandes.

Una forma eficiente de poder hallar estos números primos es mediante el método creado por Miller-Rabin, este nos dice:

Teorema 1.3. *Sea p primo,* $p-1 = 2^u k$*, donde k es impar y* $u > 0$*, sea* $a < p$*, tomaremos* $b = a^k$ *entonces* $b = \pm 1 \mod p$ *o bien* $\exists r,\ 0 < r < u$ *tal que* $b^{2^r} = -1 \mod p$.

Demostración. Por el Pequeño Teorema de Fermat sabemos que $a^{p-1} = 1 \mod p$ que esto por nuestro enunciado podemos reescribirlo de la siguiente manera $a^{k2^u} = 1 \mod p$ que es lo mismo que $b^{2^u} = (b^{2^{u-1}})^2 = 1 \mod p$. Dado que $\mathbb{Z}_p$ es un cuerpo, las únicas soluciones de la ecuación $x^2 = 1 \mod p$ son $x = \pm 1 \mod p$. Por lo tanto $b^{2^{u-1}} = \pm 1 \mod p$.

- Si $b^{2^{u-1}} = -1 \mod p$ hemos terminado, puesto que $u - 1 < u$.
- En caso contrario, se volvería a extraer la raíz cuadrada y así sucesivamente y en el peor de los casos en el que no podamos seguir extrayendo raíces llegamos a que $b^2 = \pm 1 \mod p$ que cumple el teorema.

□

La seguridad del método RSA, como ya hemos especificado, radica en la dificultad de la descomposición de n en dos números primos, seguridad que podría verse amenazada debido a los ordenadores cuánticos, un nuevo tipo de ordenadores todavía no construidos a causa de la necesidad de unas condiciones muy precisas para su funcionamiento. Este nuevo prototipo sería mucho más veloz y capaz de resolver procesos matemáticos transcendentales como podría ser dicha descomposición.

Capítulo 2

Introducción a las bases de Groebner

Con el objetivo de explorar otros posibles criptosistemas más complejos, introduciremos métodos relacionados con anillos de polinomios en varias variables con coeficientes en cuerpos finitos. Una de las herramientas más útiles en este contexto desde el punto de vista computacional son las bases de Groebner. Nuestro propósito a lo largo de este capítulo es definir y calcular de manera efectiva dichas bases. El concepto de base de Groebner está íntimamente relacoinado con el de ideal monomial, que introducimos a continuación.

Definición. Se denomina *monomio* en varias variables, $x_1, ..., x_n$, a un producto de la siguiente forma:

$$x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$$

donde todos los exponentes $\alpha_1, ..., \alpha_n$ son enteros no negativos. El *grado* del monomio x^α viene dado por la suma de los exponentes es decir:

$$\alpha_1 + ... + \alpha_n$$

Durante todo el capítulo simplificaremos la notación y a dichos exponentes los denotaremos como $\alpha = (\alpha_1, ..., \alpha_n)$ tal que $x^\alpha = x^{\alpha_1} ... x^{\alpha_n}$. Tal que si $\alpha = (0..,0)$ esto implica que $x^\alpha = 1$.

Definición. Se denomina *polinomio* f en varias variables, $x_1, ..., x_n$, con coeficientes en $\mathbb{K}$ cuerpo, a una combinación lineal finita de monomios.

$$f = \sum_\alpha a_\alpha x^\alpha \text{ con } a_\alpha \in \mathbb{K}$$

Definiremos a_α como el *coeficiente* del monomio x_α y si $a_\alpha \neq 0$ entonces $a_\alpha x^\alpha$ es un *término* de f. Utilizando la notación anterior podemos definir el *grado* de un polinomio f como el máximo $|\alpha|$ tal que el coeficiente de $a_\alpha \neq 0$.

El conjunto de los polinomios de variables $x_1, ..., x_n$ con coeficientes sobre un cuerpo, $\mathbb{K}$, se denota como $\mathbb{K}[x_1, ..., x_n]$. Añadir que cuando hablamos de la suma y el producto de dos polinomios lo que obtenemos es un nuevo polinomio.

Otra de las partes fundamentales del álgebra es el estudio de los ideales así pues vamos a definirlos.

Definición. Se dice que un subconjunto $I \subset \mathbb{K}[x_1, ..., x_n]$ es un ideal si satisface:

- $0 \in I$
- Si $f, g \in I$, entonces $f + g \in I$
- Si $f \in I$ y $h \in \mathbb{K}[x_1, ..., x_n]$ entonces $hf \in I$

Ya que daremos una mayor importancia al uso de polinomios en varias variables definamos la idea de ideal generado por un número finito de polinomios.

Definición. Sea $f_1,...,f_s$ monomios en $\mathbb{K}[x_1,...,x_n]$. Entonces nuestro conjunto:

$$\langle f_1,...,f_s\rangle = \{\sum_{i=1}^{s} h_i f_i \mid h_1,...,h_s \in \mathbb{K}[x_1,...,x_n]\}$$

Donde $\langle f_1,...,f_s\rangle$ es un ideal monomial.

Proposición 2.1. *Dos ideales monomiales son iguales si y solo si poseen los mismos elementos.*

Para la culminación de esta sección mostraremos el siguiente teorema que nos dará una propiedad fundamental de un anillo de polinomios sobre un cuerpo.

Teorema 2.1. ***Teorema de la base de Hilbert***

1. *Si I es un ideal de R entonces está finitamente generado.*
2. *(Condición de cadena ascentente) Si $I_1 \subseteq I_2 \subseteq ... \subseteq I_n \subseteq ...$ es una cadena ascendente de ideales entonces existe un N tal que $I_N = I_M$ para todo $N \leq M$.*

Su demostración se encontrará en el apéndice.

2.1. Orden monomial

La división de los polinomios de $\mathbb{K}[x]$ presenta un algoritmo cuya clave principal es el orden debido al grado de los monomios, ya que, como bien es sabido solamente podemos dividir si el grado(dividendo) es mayor que el grado(división) y posteriormente cuando grado(resto) sea mayor que el grado(división), por lo tanto a la hora de definir una división de polinomios en más de una variable es importante precisar un orden en el conjunto de los monomios para poder volver a hablar del orden necesario y del algoritmo de la división.

Luego definamos qué es un orden monomial.

Definición. Definimos un orden monomial, $>$, en K sobre las variables $[x_1,...,x_n]$ como una relación en $\mathbb{Z}^n_{\geq 0}$ o equivalentemente una relación sobre el conjunto de monomios x^α, $\alpha \in \mathbb{Z}^n_{\geq 0}$ satisfaciendo:

- $>$ es un orden total en $\mathbb{Z}^n_{\geq 0}$
- Si $a > b$ y $\gamma \in \mathbb{Z}^n_{\geq 0}$ entonces $a+\gamma > b+\gamma$.
- $>$ es un buen orden en $\mathbb{Z}^n_{\geq 0}$. Esto significa que cada subconjunto distinto del vacío de $\mathbb{Z}^n_{\geq 0}$ tiene un elemento minimal.

Para entender correctamente las bases de Groebner veremos el siguiente lema:

Lema 2.1. *Una relación de orden $>$ en $\mathbb{Z}^n_{\geq 0}$ es un buen orden si y solo si existe una secuencia estrictamente decreciente en $\mathbb{Z}^n_{\geq 0}$ que termina.*

Demostración. Probémoslo por contraposición, es decir, supongamos que $>$ no es un buen orden si y solo si existe una secuencia estrictamente decreciente en $\mathbb{Z}^n_{\geq 0}$.

Veamos la primera implicación; si $>$ no es un buen orden, por la tercera condición de la definición de un orden monomial podemos decir que algún subconjunto no vacío $S \subset \mathbb{Z}^n_{\geq 0}$ no tiene un elemento minimal. Tomemos $\alpha(1) \in S$. $\alpha(1)$ no es el elemento minimal esto implica que existe un $\alpha(2) \in S$ tal que $\alpha(1) > \alpha(2)$, pero aquí ocurrirá lo mismo, es decir, existirá un $\alpha(3) \in S$ tal que $\alpha(1) > \alpha(2) > \alpha(3)$. Continuándolo crearemos una cadena infinita que decrece estrictamente.

Por otro lado, si tenemos una secuencia infinita $\{\alpha(1),\alpha(2),...\}$ esto es un subconjunto no vacío de $\mathbb{Z}^n_{\geq 0}$ que no posee elemento minimal, por lo tanto, $>$ no será un buen orden. □

Gracias a este lema podremos posteriormente asumir la terminación de varios algoritmos, como puede ser el de la división.

Así pues, existen gran variedad de órdenes monomiales como pueden ser el orden lexicográfico, orden graduado reverso-lexicográfico, el orden graduado lexicográfico... El orden que nosotros utilizaremos y por consiguiente el que más nos interesará será el orden lexicográfico.

Definición. Definiremos el orden monomial lexicográfico de la siguiente manera; sea $\alpha = (\alpha_1, ..., \alpha_n)$ y $\beta = (\beta_1, ..., \beta_n) \in \mathbb{Z}^n_{\geq 0}$. Diremos que $\alpha <_{lex} \beta$ si y solo si

$$\begin{cases} \alpha_1 < \beta_1 \text{ ó} \\ \alpha_1 = \beta_1 \text{ y } \alpha_2 < \beta_2 \text{ ó} \\ ... \end{cases}$$

Veamos pues un ejemplo para ver como funciona dicho orden:

Ejemplo 2.1. *Sea $f_1 = xy+1$ y $f_2 = y^2 - 1$ tal que definimos el orden lexicográfico $x > y$ luego escribiendo nuestros polinomios como un vector donde la primera coordenada es el grado de la x y la segunda el grado de la y, nos queda:*

$$\begin{array}{c} \textit{Para } f_1 \longrightarrow \alpha = (1,1) \\ \textit{y para } f_2 \longrightarrow \beta = (0,2) \end{array}$$

Como hemos definido que poseemos un orden lexicográfico donde $x > y$, vemos que $\alpha > \beta$

Para esto usaremos la siguiente terminología:

Definición. Sea $f = \sum_\alpha a_\alpha x^\alpha$ un polinomio distinto de cero en $\mathbb{K}[x_1, ..., x_n]$ y sea $>$ un orden monomial, definiremos:

- El coeficiente principal como el $lc(f)$: el coeficiente del término de mayor grado.
- El monomio principal como el $lm(f)$: el monomio de mayor grado sin el coeficiente.
- El término principal como el $lt(f) = lc(f)lm(f)$

Dicho esto, ya poseemos una manera de ordenar los monomios de nuestros polinomios luego vamos ahora a buscar un algoritmo de la división en varias variables sobre el cuerpo $\mathbb{K}[x_1, ..., x_n]$ dicho algoritmo buscará extender el existente en $\mathbb{K}[x]$. En general el objetivo es dividir $f \in \mathbb{K}[x_1, ..., x_n]$ entre $f_1, ..., f_s \in \mathbb{K}[x_1, ..., x_n]$, esto se expresará de la siguiente manera:

$$f = a_1 f_1 + a_2 f_2 + ... + a_s f_s + r$$

donde los cocientes son $a_1, ..., a_s$, y r es el resto en $\mathbb{K}[x_1, ..., x_n]$, debemos tener cuidado a la hora de elegir dicho resto y aquí será donde daremos uso al orden monomial definido, fundamentalmente.

Teorema 2.2. ***Algoritmo de la división en*** $\mathbb{K}[x_1, ..., x_n]$**.** *Fijando un orden monomial $>$ en $\mathbb{Z}^n_{\geq 0}$ y sea $F = (f_1, ..., f_s)$ una s-tupla ordenada del conjunto de polinomios $\mathbb{K}[x_1, ..., x_n]$. Entonces cada $f \in \mathbb{K}[x_1, ..., x_n]$ se puede escribir como:*

$$f = a_1 f_1 + a_2 f_2 + ... + a_s f_s + r$$

donde $a_i, r \in \mathbb{K}[x_1, ..., x_n]$ y o el resto, r es 0 o es una combinación lineal con coeficientes en $\mathbb{K}$, sus monomios, ninguno puede ser divisible entre $lt(f_i)$ con $i = 1, ..., s$.
Además si $a_i f_i \neq 0$, entonces

$$\textit{grado(f)} \geq \textit{grado } (a_i f_i).$$

Veamos un ejemplo de dicho algoritmo:

Ejemplo 2.2. *Queremos dividir el polinomio $f = x^2y + xy^2 + y^2$ entre $f_1 = y^2 - 1$ y $f_2 = xy - 1$ usando el orden monomial lexicográfico $x > y$. Frente a dicho orden ya tenemos a nuestros polinomios en varias variables ordenados, además elegimos dividir f entre f_1 y posteriormente entre f_2.*

Dividimos pues $x^2y + xy^2 + y^2$ entre $y^2 - 1$, vemos que el $lt(y^2 - 1) = y^2$ luego debemos buscar los monomios de $x^2y + xy^2 + y^2$ que puedan ser divididos por y^2, el primero que podemos dividir será xy^2, obtendremos un resto y continuando con el mismo mecanismo llegaremos a que dicha división posee un cociente que es $x + 1$ y un resto $x^2y + x + 1$. Observamos que ningún término de nuestro resto puede volver a ser dividido entre el $lt(y^2 - 1)$. Luego ahora tomamos dicho resto y lo dividimos entre f_2, con la misma mecánica y su cociente será x y su resto $2x + 1$.

Recopilando todo tenemos:

$$x^2y + xy^2 + y^2 = (x + 1)(y^2 - 1) + x(xy - 1) + 2x + 1$$

donde podemos notar que ninguno de los monomios del resto puede ser divisible entre $lt(f_i)$ con $i = 1, 2$.

Veamos pues, el principal resultado de ideales monomiales de $\mathbb{K}[x_1, ..., x_n]$, que nos dice que son finitamente generados.

Lema 2.2. ***Lema de Dickson.*** *Sea $I = \langle x^\alpha : \alpha \in A \rangle \subset \mathbb{K}[x_1, ..., x_n]$ un ideal monomial . Entonces I puede ser escrito de la siguiente forma $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$ donde los $\alpha(i) \in A$. En particular, I tiene una base finita.*

2.2. Bases de Groebner

Sea $R = \mathbb{K}[X]$ el anillo de polinomios sobre un cuerpo $\mathbb{K}$, donde $X = \{x_1, ..., x_m\}$. Dado un orden monomial y una n-tupla ordenada de polinomios $\{f_1, ..., f_n\}$, podemos calcular el resto r de la división multivariante de $f \in R$. Pero ¿Este resto cuando a f lo dividimos entre entre los $f_1, ..., f_s$ ordenados de diferente forma se mantiene?.

Ejemplo 2.3. *Tomando el mismo ejemplo de antes donde $f = x^2y + xy^2 + y^2$, $f_1 = y^2 - 1$ y $f_2 = xy - 1$, veamos qué ocurriría si decidiésemos dividir f primero entre f_2 y posteriormente entre f_1. Veámoslo.*

Tomamos nuestro polinomio $f = x^2y + xy^2 + y^2$ y comenzamos dividiéndolo entre $xy - 1$, al igual que antes poseeremos un cociente que será $x + y$ y un resto $x + y + y^2$ que no puede ser dividido por el $lt(xy - 1)$. Luego tomamos dicho resto y lo dividimos entre $f_1 = y^2 - 1$, su cociente será 1 y su resto $x + y + 1$.

Recopilando tenemos:

$$x^2y + xy^2 + y^2 = (x + y)(xy - 1) + (y^2 - 1) + x + y + 1$$

Comparando las dos expresiones obtenidas que han resultado de dividir f entre f_1, f_2 vemos que al variar el orden de los divisiores, las f_i, el resto nos varía. Esto nos indica que el algoritmo de la división no nos caracteriza de manera única el resto r y respondiendo a nuestra pregunta, si cambiamos el orden de los f_i este variará.

Nosotros conocemos que en el caso de un polinomio de una sola variable:

$$f \in \langle g \rangle \Leftrightarrow \text{el resto de la división Euclídea de } f \text{ por } g \text{ es } 0.$$

Ahora estos conceptos variarán puesto que si tenemos un ideal $I = \langle f_1, ..., f_n \rangle \in R$, y al dividir $r = 0$, esto nos dice que $f \in I$. Pero el recíproco no es cierto en general ya que los órdenes monomiales a veces impiden que este pueda ser cancelado, ilustrémoslo en un ejemplo.

Ejemplo 2.4. *Consideremos los polinomios sobre $\mathbb{Q}[x,y]$ y sean $f_1 = xy - y$, $f_2 = x + 1$ y $f = xy$. Tomemos el orden lexicográfico con $x > y$. El resto de dividir f entre f_1, f_2 es $r = y \neq 0$ sin embargo se puede observar que f pertenece al ideal $\langle f_1, f_2 \rangle$. La pregunta sería que es lo que sucede; así $r = y \in \langle f_1, f_2 \rangle$. Podemos observar que $r = f - f_1 = xy - xy + y = y$ pero no existe ningún monomio líder en los polinomios tal que pueda cancelarlo. Esto implica que en verdad, nuestro resto r es 0 módulo f, f_1, f_2.*

Luego para poder conseguir mediante la división multivariante de $f_1, ..., f_n$ que un resto no nulo implique la no pertenencia al ideal $I = \langle f_1, ..., f_n \rangle$ debemos poder eliminar todos los términos líderes de los elementos de I por medio de los términos líderes de $f_1, ..., f_n$. Así pues,

Definición. Dado un orden monomial, $>$ y un ideal $I \in R$ diremos que $\{f_1, ..., f_n\} \subset R$ es base de Groebner de I para dicho orden monomial si cumple que:

$$\langle lt(f_1), ..., lt(f_n) \rangle = \langle lt(I) \rangle$$

donde $lt_>(I) = \{lt_>(f) | f \in I\}$

Dicho de una manera mas informal, si tenemos un conjunto $\{g_1, ..., g_m\} \subset I$ es una base de Groebner del ideal I si y solo si el lt de cada elemento de I es divisible entre cada uno de los $lt(g_i)$ $\forall i \in \{1, ..., m\}$.

Así pues, adaptando dicha definición afirmamos que efectivamente una base de Groebner genera el ideal I de R en el siguiente sentido. Un conjunto $F = \{f_1, ..., f_m\} \subset I$ genera el ideal I si todo elemento $f \in I$ puede escribirse como combinación de los elementos de F, es decir,

$$f = \sum_{i=1}^{m} h_i f_i$$

para ciertos $h_1, ..., h_m \in R$. Probaremos que todo ideal I del anillo de polinomios R admite una base de Groebner y que toda base de Groebner genera I.

Proposición 2.2. *Dado un orden monomial, $>$, y un ideal $I \subset R$, existe una base de Groebner G de I y esta G es un sistema de generadores de I.*

Para su demostración necesitamos demostrar el siguiente teorema:

Teorema 2.3. *Si $G = \{g_1, ..., g_n\}$ es una base de Groebner del ideal $I \subset R$ para el orden monomial, $>$, $f \in R$ $\Rightarrow$ Existe un único $r \in R$ que cumple las siguientes propiedades:*

- *Ningún término de r se puede dividir por $lt_>(g_i)$*
- *Existe una $g \in I$ tal que $f = g + r$.*

Demostración. Sea supuesto que $\exists r, r' \in R$ tal que $f = g + r$ y $f = g' + r'$, donde $g, g' \in I$.

$$r - r' = g - g' \in I$$

Si $r - r' \neq 0$ esto implicaría la existencia de algún $lt_>(g_i)$ con $1 \leq i \leq n$ que dividiría a $lt_>(r - r')$, dicho hecho no es posible ya que ninguno de ellos divide ni a r ni a r'. □

Así pues, habíamos visto en los ejemplos 2,2 y 2,3 que si queremos dividir un f entre $\{f_1, ..., f_n\}$ y decidimos variar el orden de los f_i a la hora de realizar la división, el resto que obtenemos es distinto, pero vemos que esto no ocurre siempre, puesto que vemos en esta proposición que el resto queda únicamente determinado cuando dividimos por una base de Groebner.

Y gracias a este teorema ya podemos demostrar la proposición 2.2

Demostración. Por el teorema de la base de Hilbert podemos asegurar que existe un sistema de generadores finitos $h_1,...,h_s$ para el ideal $\langle lt_>(I)\rangle$. Como $\langle lt_>(I)\rangle$ está generado por los términos líderes de I podemos encontrar polinomios $\{g_1,...,g_s\} \in I$ tales que:

$$h_i \in \langle lt_>(g_1),...,lt_>(g_s)\rangle \text{ con } 1 \leqslant i \leqslant s$$

Esto significa que:

$$\langle lt_>(I)\rangle = \langle h_1,...,h_s\rangle \subset \langle lt_>(g_1),...,lt_>(g_s)\rangle \subset \langle lt_>(I)\rangle$$

de donde podemos deducir que $\langle lt_>(I)\rangle = \langle lt_>(g_1),...,lt_>(g_s)\rangle$ y por lo tanto el conjunto de polinomios $\{g_1,...,g_s\}$ es una base de Groebner. Sea definido por construcción:

$$f = \sum_{i=1}^{m} h_i f_i + r$$

Sabemos que $f \in I$, que el sumatorio también pertenece a I y por lo tanto se deduce que $r \in I$. Así tenemos dos posibilidades:

- $r = 0$ con lo que concluiríamos la demostración.
- $lt(r) \in lt(I) = lt(f_i)$ que es contradicción puesto que por teoría sabemos que ningún monomio de r es múltiplo de $lt(f_i)$.

□

El siguiente resultado define la propiedad fundamental de las bases de Groebner. Para enunciarlo adecuadamente, denotaremos los restos de la división multivariante de f por G para un determinado orden monomial como:

$$\overline{f}^G = r.$$

Teorema 2.4. *Si G es una base de Groebner del ideal $I \subset R$ entonces*

$$f \in I \;\; \Leftrightarrow \overline{f}^G = 0.$$

Demostración. Si el resto es 0, entonces ya tenemos lo que buscábamos que $f \in I$. La implicación opuesta viene porque sea $f \in I$ entonces $f = f + 0$, que satisface las dos condiciones de la proposición anterior, luego se obtiene que 0 es el resto de dividir f entre la base de Groebner G. □

Este resultado nos ayudará a responder a las siguientes cuestiones:

- Determinar si un polinomio f pertenece a un ideal
- Si pertenece, hallar los $s_i \in R \;\; i \in \{1,...,n\}$ tal que $s_1 f_1 + ... + s_n f_n = f$

Así pues detallaremos la siguiente condición; sea $lt(I) = \langle lt(g_1),...,lt(g_n)\rangle$ y sea $f \in R$ ninguno de los monomios de r puede ser múltiplo de ningún $LT(g_i)$.

Y definimos:

$$\langle x^\alpha : x^\alpha \text{ no es múltiplo de } lt(g_i)\rangle \simeq \frac{R}{I}$$

Esto representa un isomorfismo respecto a un espacio vectorial.

2.3. Criterio de Buchberger

Hemos dado una serie resultados valiosos de bases de Groebner que nos permiten afirmar la existencia y nos dan una serie de propiedades una vez conocida dicha base. Ahora, nuestra finalidad va a ser la capacidad de construir nosotros la base de Groebner. Para ello necesitamos plantear qué es el *mcm* de varios polinomios.

Definición. Sea $f,g \in \mathbb{K}[x_1,...,x_n]$ y dichos polinomios son distintos de cero.
Entonces si el grado(f)=α y el grado(g)= β, sea $\gamma = (\gamma_1,...,\gamma_n)$ donde los $\gamma_i = max(\alpha,\beta)$ para cada i. Llamaremos a x^γ el mínimo común múltiplo de $lm(f)$ y $lm(g)$ que será denotado como $x^\alpha = mcm(lm(f),lm(g))$.

Luego ahora, ya tenemos todo listo para poder definir lo que son los *S*-polinomios.

Definición. Sean f,g dos polinomios no nulos de *R* y establecemos un *orden monomial*, $>$ y definimos $x^\alpha = mcm(\{lm_>(f),lm_>(g)\})$. Definiremos el *S*-polinomio de f,g como:

$$S(f,g) = \frac{x^\alpha}{lt_>(f)} \cdot f - \frac{x^\alpha}{lt_>(g)} \cdot g$$

Este resultado nos será útil para ver si algo es o no base de Groebner, debido a que si todos los *S*-polinomios de nuestro ideal, *I*, son iguales a cero entonces afirmaremos que poseemos una base de Groebner. Formalicemos dicha información.

Teorema 2.5. ***Criterio de Buchberger.*** *Fijado un orden monomial, $>$, sea I ideal de R.*

Un sistema de generadores $G = \{g_1,...,g_n\}$ de I es base de Groebner $\Leftrightarrow$ $\overline{S(g_i,g_j)}^G = 0 \ \ \forall i,j: \ \ i \neq j$

Luego podemos llegar a asegurar que podemos construir una base de Groebner en un número finito de pasos, gracias a los S-polinomios. Veamos con un ejemplo de como podemos construir a partir del algoritmo de Buchberger una base de Groebner.

Ejemplo 2.5. *Sea $f_1 = xy - y$, $f_2 = x+1$ sobre el cuerpo $\mathbb{Q}[x,y]$. Tal que generan un ideal $I = \langle f_1, f_2 \rangle$, tomamos el orden monomial lexicográfico $x > y$.*

Veamos para empezar que I no es base de Groebner, para ello tomamos los $lt(f_1) = xy$ y $lt(f_2) = x$. Luego el mcm $= xy$ luego

$$S(f_1,f_2) = \frac{xy}{xy} f_1 - \frac{xy}{x} f_2 = f_1 - y f_2 = (xy - y) - y(x+1) = -2y \neq 0$$

Ahora debemos dividir este resto del S-polinomial entre f_1, f_2 como no es cero respecto a los ideales, por el criterio anterior sabemos que no poseemos una base de Groebner, pero queremos generarla, la primera idea que podemos tener consiste en añadir algo a nuestro ideal de manera que se nos expanda la base pero si añadimos un polinomio sin cierto rigor seguramente obtendremos elementos redundantes. Luego qué debemos añadir, pues lo más ventajoso y según el criterio de Buchberger ya que queremos que los S-polinomios sean ceros será añadir el resto que nos ha quedado en nuestro S-polinomio. Es decir ahora tomamos $f_3 = -2y$ y creamos el nuevo conjunto $F = \{f_1, f_2, f_3\}$. Y como es de esperar volveremos a realizar todos los S-polinomial para ver si estos son 0 respecto a nuestro ideal.

$$\begin{array}{lcll} \overline{S(f_1,f_2)}^F & = & 0 & \\ \overline{S(f_1,f_3)}^F & = & 2y & = \ 0 \\ \overline{S(f_2,f_3)}^F & = & -2y & = \ 0. \end{array}$$

En nuestro caso, todos los S-polinomios lo son, esto implica por el Criterio de Buchberger que $F = \{f_1, f_2, f_3\}$ es una base de Groebner.

Por lo tanto, recapitulemos y creemos un algoritmo que nos permita construir siempre bases de Groebner en un número finito de pasos.

Tomamos un ideal $I = \langle f_1, ..., f_n \rangle$, sobre un cuerpo con un determinado orden monomial, construimos sus S-polinomios, si estos son cero, ya poseemos una base de Groebner, si no lo son debemos añadir el resto de nuestro S-polinomio, que lo llamaremos s y construir un nuevo $F = \{f_1, ..., f_n, s\}$. Y ahora volveríamos a comenzar reiterando lo dicho hasta que consiguiéramos que todos los S-polinomios fueran 0, donde nuestro algoritmo se terminaría. A este algoritmo se le denomina algoritmo de Buchberger. Pero establezcamos pues, un teorema para todo lo que acabamos de decir.

Teorema 2.6. *Si $I = \langle f_1, ..., f_n \rangle \in R$ es un ideal distinto de $0 \Rightarrow$ se puede construir una base de Groebner para cada I con un determinado orden monomial, gracias al algoritmo de Buchberger.*

Demostración. Comenzamos con nuestro ideal, que será el posible candidato a base de Groebner, llamémoslo $G = \langle f_1, ..., f_n \rangle \in R$, tal que generaremos sus S-polinomios teniendo dos posibles resultados:

- $S(f_i, f_j) = 0$ con $i \neq j$, donde hemos acabado la demostración porque poseeremos una base de Groebner.
- $S(f_i, f_j) \neq 0 (= s)$ con $i \neq j$, donde lo que hacemos es incluir esta s a nuestra posible base $G = \{G\} \cup \{S^G(f_i, f_j)\}$

Los polinomios que vamos añadiendo van reduciendo el grado puesto que es trivial que el grado del *lt* de los S-polinomios va siendo cada vez menor. Hasta que finalmente queda completamente reducido y no podemos llegar a incluir ningún otro polinomio. □

El algoritmo de Buchberger nos ha permitido la creación de una base de Groebner de una manera rudimentaria y laboriosa, es decir, es un buen método pero no es práctico a la hora de generarla porque si nos damos cuenta, podemos incluir a dicha base otros elementos tal que su S-polinomio en dicha base siga siendo cero. Luego esto quiere decir que a lo mejor nosotros ya poseemos una base de Groebner pero con un número altamente reiterado de generadores, es decir, elementos redundantes que no son útiles y que pueden ser suprimidos y de los que podemos considerar su reducción, veamos como debemos ejecutarlo.

Teorema 2.7. *Sea G una base de Groebner para el ideal I. Sea $g \in G$ tal que $lt_>(g) \in \langle lt_>(G) \setminus \{g\} \rangle \Rightarrow G \setminus \{g\}$ es base de Groebner del ideal $I \subset R$ y del orden dado.*

Demostración. Por definición de base de Groebner sabemos que si G es base de Groebner de un ideal I, $\langle lt_>(G) \rangle = \langle >(I) \rangle$ pero por hipótesis de nuestra proposición tenemos que $\langle lt_>(G) \rangle = \langle lt_>(G) \setminus \{g\} \rangle$. Luego por lo tanto $(G) \setminus \{g\}$ es base de Groebner de $I \in R$. □

De esta manera nosotros podemos proceder a la eliminación de los términos que no tienen valor, pero además si reajustamos las constantes de los *lt* imponiendo que dichos polinomios posean como coeficiente líder el 1 llegaremos a la construcción de una base de Groebner minimal.

Definición. Sea $I \subset R$ un *ideal* no nulo y definimos un orden monomial, $>$. Sea una base de Groebner G de I. Diremos que esta es una base de Groebner minimal si:

- $lc(g) = 1 \ \ \forall g \in G$
- $\forall g \in G$ se tiene que $lt_>(g) \notin \langle lt_>(G \setminus \{g\}) \rangle$.

Desafortunadamente, un ideal puede tener muchas bases de Groebner minimales:

Ejemplo 2.6. *Sea definido un* orden monomial lexicográfico *con $x < y$ sean $f_1 = y + ax + 1$ y $f_2 = x$ con $a \in \mathbb{Q}$, es fácil ver que nuestra base es una base de Groebner minimal, pero como $a \in \mathbb{Q}$ existirán infinitas.*

Luego debemos profundizar más a la hora de buscar la existencia de una única base de Groebner.

Definición. Llamamos *base de Groebner reducida* de un ideal $I \subset R$ respecto a un orden monomial, $>$, a una base tal que:

- $lc(g) = 1 \ \ \forall g \in G$
- $\forall g \in G$ ningún monomio de g pertenece a $\langle lt_>(G \setminus \{g\})\rangle$.

Tener en cuenta que una base reducida también cumple la definición de base minimal por lo tanto cualquier base reducida será base minimal. Y además para un determinado orden monomial solo existirá una base de Groebner reducida que como consecuencia también será minimal.

Proposición 2.3. *Sea $I \neq \{0\} \subset R$ un ideal. Sea definido un orden monomial, $>$, entonces para dicho orden solo existirá una única base de Groebner reducida.*

Demostración. Sea G una *base de Groebner minimal* de I. Nosotros diremos que $g \in G$ es un elemento reducido de G siempre y cuando ningún monomio de g está en $\langle lt(G) \setminus \{g\}\rangle$. Nuestra propuesta pues es modificar G hasta que todos los elementos sean reducidos.
Primero observemos que si g es reducido en G, entonces g es también reducido por cualquier otra base de Groebner minimal de I que contenga a g y posea el mismo conjunto de lt. Esto se sigue porque la definición de elemento reducido engloba a la de lt.

Ahora, dado $g \in G$, sea $g' = \overline{g}^{G-g}$ y sea el conjunto $G' = (G \setminus \{g\}) \cup \{g'\}$. Afirmamos que G' es base de Groebner minimal de I. Para verlo, notemos que $lt(g') = lt(g)$, porque cuando dividimos g entre $G \setminus g$, el $lt(g)$ hace de resto puesto que no es divisible por ningún elemento de $(lt(G) \setminus \{g\})$. Lo que nos lleva a mostrar que $\langle lt(G')\rangle = \langle lt(G)\rangle$. Claramente G' esta contenido en I, luego es una base de Groebner minimal de I. Notemos también que g' es reducido sobre $G^{'}$ por construcción.

Ahora, tomemos los elementos de G y apliquemos el proceso hasta que estén reducidas. La base de Groebner podría cambiar pero la primera observación muestra que cada elemento reducido nunca cambia su lt. Luego hemos acabado con la reducción de una base de Groebner.

Veamos ahora la unicidad, supongamos que G y $\overline{G}$ son bases de Groebner reducidas para un I. En particular, G y $\overline{G}$ son bases de Groebner minimales, luego esto implica que sus lt son iguales.
Entonces dado $g \in G$ existirá un $\overline{g} \in \overline{G}$ tal que sus lt sean iguales. Si podemos mostrar que $g = \overline{g}$ entonces obtendremos la unicidad.
Para verlo consideremos $g - \overline{g}$. Esto se encuentra en I y como G es una base de Groebner, esto implica $\overline{g - \overline{g}}^{G} = 0$. También sabemos que $lt(g) = lt(\overline{g})$. Entonces estos términos se cancelan en $g - \overline{g}$, y que el resto no es divisible entre los $lt(G) = lt(G)$ ya que G y G' son bases reducidas. Así $\overline{g - \overline{g}}^{G} = g - \overline{g}$ y entonces, $g - \overline{g}$ esto es igual a cero y se sigue el teorema.

□

Así las bases de Groebner, con su algoritmo de Buchberger para hallarlas son muy utilizadas en todo lo que a computación se refiere puesto que nos sirven para saber si un polinomio se encuentra en un determinado ideal, para resolver un conjunto de ecuaciones, para parametrizar ecuaciones... Veamos en el capítulo siguiente alguna otra utilidad de dicha base.

Capítulo 3

Sistemas criptográficos polinómicos. Polly Cracker. El problema Inverso.

En esta sección hablaremos de dos tipos de criptosistemas de clave pública, los llamados criptosistemas de Polly Cracker y el inverso del polinomio. Estos se caracterizan por estar apoyados en la teoría de las bases de Groebner.

3.1. Criptosistema de Polly Cracker

Los sistemas de Polly Cracker son un tipo de criptosistema que se desarrolla sobre un cuerpo de múltiples variables y de p elementos $\{0,1\}$, $R = \mathbb{F}_{p^r}[X]$, donde $X = \{x_1, ..., x_n\}$. Generalmente, nuestra $p = 2$ luego nuestro criptosistema vendrá dado sobre $R = \mathbb{F}_{2^r}[X]$
Su clave pública viene dada por el conjunto de polinomios $f_i \in R \ \ \forall i \in \mathbb{Z}$ que generan un ideal I, es decir $I = \langle f_1, ..., f \rangle$ y su clave privada originalmente es proporcionada por $\alpha \in V(f_1, ..., f_n)$ donde la notación $V(f_1, ..., f_n)$ viene a darnos los ceros del ideal y así dicho α será algún cero del ideal. Como es de esperar la dificultad de este problema radica en hallar las soluciones de dicho conjunto.

Veamos ahora cual es su funcionamiento; el problema de Polly Cracker originario es retomando las figuras de Bob y Alicia usadas anteriormente; Bob desea enviar un mensaje a Alicia, luego Alicia toma un conjunto de polinomios F que generaran el ideal I, y publicará dichos polinomios como su clave pública. Tal y como hemos dicho antes Alicia conocerá un cero de dicho ideal, $\alpha \in V(I)$ que es lo mismo que decir que para todo polinomio f de nuestro ideal $f(\alpha) = 0$, y este cero vendrá a ser la clave privada que solo ella poseerá.
Luego sea $m \in \mathbb{F}_{2^r}$ el mensaje a cifrar por Bob. Su codificación vendrá dada por $c = m + \Sigma h_i(X) \cdot F_i(X)$ donde los $h_i \in R$ serán polinomios aleatorios que tomará Bob de la clave pública que Alicia ha dado a conocer, es decir los $h_i \in I$.

Para desencriptar dicho mensaje, Alicia que ya posee un cero del ideal I, llamémoslo α, lo evaluará en dicho polinomio es decir; $c(\alpha)$ y así podremos desencriptar el mensaje m. Esto consiste únicamente en sustituir, sabiendo que los $f_i(\alpha) = 0$.

Una alternativa dada a estos criptosistemas fue propuesta por Barkee, que se encuentra basada completamente en las bases de Groebner. Su funcionamiento viene dado gracias a que nosotros poseemos una clave pública que es nuestro ideal I, de dicho ideal conocemos una base de Groebner G que será su clave privada y así manteniendo la mecánica del criptosistema de Polly Cracker obtendremos que Bob enviará un mensaje a Alicia m, eligiendo un polinomio aleatorio h tal que calcularemos $c = m + h$, cuando Alicia reciba dicho mensaje lo reducirá respecto a su base de Groebner y ya podrá obtener el mensaje.

Ejemplo 3.1. *Un ejemplo de este tipo de criptosistemas puede ser el 3-coloreado de un grafo. Para ello sea* $\Gamma = \Gamma(V,E)$ *un grafo con vértices* V *donde* $V \in \{1,...,n\}$ *y con aristas* $E \in \{\{i,j\}, 1 \leq i < j \leq n\}$*. Lo que queremos realizar es colorear los vértices del grafo de manera que todos los vértices posean algún color con la condición de que vértices adyacentes no puedan ser coloreados del mismo color, donde supongamos que se conoce una 3-coloración del grafo, esto corresponde en lenguaje matemático a poseer una aplicación* $\phi : V \to \{0,1,2\}$ *tal que se verifica que si* $\{u,v\} \in E \ \Rightarrow \phi(u) \neq \phi(v)$*.*

Iniciaremos nuestro problema plasmando esta aplicación de manera polinómica, de manera que definiremos el conjunto de polinomios F_0, F_1, F_2 *sobre las variables* $x_{i,k}$*. Para continuar deberemos imponer la siguiente condición:*

$$\{x_{i,k}\}_{i \in V, k \in \mathbb{Z}_3} \text{ tal que } \phi(i) = k \ \Leftrightarrow x_{i,k} = 1, \text{ en el caso opuesto } x_{i,k} = 0.$$

Luego vemos que hemos impuesto que nuestras variables $x_{i,k} \in \mathbb{Z}_2$*.*

Ahora como queremos encontrar una solución a este problema debemos ir asignando una serie de cláusulas para llegar a nuestra solución óptima.

- *Un vértice no puede ser pintado con más de un color, que escribiéndolo como conjunto de polinomios será* $F_0 = \{x_{i,k}X_{i,j} \mid i \in V, k \neq j, \{k,j\} \in \mathbb{Z}_3\}$*.*
  Esta condición impone que $x_{i,k}x_{i,j} = 0$ *como nuestras variables pertenecen a* $\mathbb{Z}_2$*, esto implicará que al menos una de ellas ha de ser 0, es decir no estar coloreada por el color que indica la variable, es decir, que solo pueda ser pintada con un color. Un caso contradictorio seria el vector* $(1,1,1)$*, puesto que al multiplicar nos daría en todos los casos 1.*

- *Un vértice debe estar pintado con al menos un color, es decir,* $F_1 = \{\sum_{i \in \mathbb{Z}_3} X_{u,i} + 1 \mid u \in V\}$*.*
  Es decir en nuestro caso buscamos que $x_{i,1} + x_{i,2} + x_{i,3} + 1 = 0$ *que al encontrarse nuestras variables en* $\mathbb{Z}_2$ *solo cabe la posibilidad de que una de las variables sera 1 o las tres sean 1, si las tres son uno deja de cumplir la primera condición, luego solo puede ocurrir que una variable sea de un color.*

Por lo tanto, si unimos ambas condiciones vemos que lo que juntas imponen es que una variable debe ser coloreada por un único color. Sigamos con las demás cláusulas.

- *Por último, tal y como nos muestra el planteamiento de nuestro problema nos faltara imponer una condición sobre los vértices adyacentes; si dos vértices son vecinos sus colores han de ser distintos:* $F_2 = \{x_{i,k}x_{j,k} \mid \{i,j\} \in E\}$*.*
  Aquí estamos afirmando que $x_{i,k}x_{j,k} = 0$ *es decir que si dos vértices se encuentran ligados por una arista no pueden poseer el mismo color, es decir una de las variables ha de ser 0.*

La clave pública de dicho problema es $F = F_0 \cup F_1 \cup F_2$*, esto quiere decir que encontrar una solución a este problema de 3-coloración viene dado por hallar algún* α *que sea cero de nuestro conjunto* F*, este* α *será la clave pública de Alicia y el conjunto* F *vendrá a ser su clave pública. Así, este problema se podrá complicar cuando existen muchas soluciones que lo hacen cierto y dichas soluciones son arduas de encontrar.*

Otro problema que nos podemos plantear es el denominado problema del *grafo perfecto*, para ello primero daremos una serie de definiciones para su mejor comprensión:

Definición. En un grafo denominaremos clique es un conjunto de vértices V tal que para todo par de vértices de V existe una arista que los conecta.

Definición. Llamaremos grafo perfecto a un grafo en el que el número cromático de cada subgrafo inducido es igual al mayor clique de ese subgrafo.

Ejemplo 3.2. *Luego el problema del grafo perfecto nos dice, tenemos de nuevo un grafo* $\Gamma = \Gamma(V,E)$ *con vértices V donde* $V \in \{1,...,n\}$ *y aristas* $E \in \{\{i,j\}, 1 \leq i < j \leq n\}$*, ahora debemos tomar un subconjunto del conjunto V de manera que si nosotros poseemos un vértice que no pertenece a dicho subconjunto, ese vértice ha de ser unido únicamente a uno de los vértices que si pertenezcan al subconjunto, de manera matemática sería tomamos un subconjunto* $V' \subset V$*, tal que siendo* $N(u) = \{u\} \cup \{v \in V \mid \{u,v\} \in E\}$*, debemos imponer que* $A = \#(N(u) \cap V') = 1$.

De la misma forma que antes, volvemos a definir nuestra aplicación, que en este caso sería $x : V \to \{0,1\}$ *tal que si* $\{u\} \in E \ \to x(u) = 1$ *si* $u \in V'$ *y por el contrario* $x(u) = 0$ *si u no pertenece a* V'.

Ahora de nuevo vamos a definir conjuntos de polinomios pero variando las pautas, las que debemos imponer ahora se realizarán sobre A y serán las siguientes:

- *Asignaremos que* $A \leq 1$*. Es decir al unir dos vértices al menos uno ha de pertenecer a* V'*,* $\{x_u x_v \mid \{u,v\} \in E\}$*, así buscamos que* $x_u x_v = 0$ *como sabemos que si* x_u *pertenece a* V' *da cero, ya lo tenemos.*
- *Por otra parte queremos que A sea impar, es decir* $\{\sum_{v \in N(u)} x_v + 1 \mid u \in V\}$*, esto quiere decir que tomaremos todas las aristas adyacentes a un vértice v cuando sean un número impar, por ejemplo tomamos un grafo donde* $V = \{1,...,7\}$ *sea* $V' = \{1,2,5,7\}$ *y tenemos como aristas el conjunto* $E = \{(1,2),(1,5),...\}$ *luego al hacer esta suma sera* $x_1 + x_2 + x_5 + 1 = 0$ *en* $\mathbb{Z}_2$ *y vemos que A es impar.*

Así juntando ambas condiciones obtendremos que A obligatoriamente ha de valer 1 lo que implica que todos los vértices de V' *estarán unidos una única vez a un vértice de V que no pertenezca a* V'*, así esto será nuestra clave privada a la que para convertirla en la clave pública se producirán muchas más conexiones entre los vértices no pertenecientes a* V'*, creando pues nuevas aristas de manera que no dejemos ver cual era la clave privada.*

Este problema posee la dificultad en saber encontrar nuestro conjunto V' *que gracias a nuestra clave pública queda totalmente resguardado del atacante.*

3.2. Criptosistemas del inverso del polinomio

Aunque existen muchos tipos de sistemas multivariantes nos vamos a centrar en la encriptación asimétrica multivariante.

Intentaremos unificar las notaciones tanto como sea posible definiendo posteriormente los aspectos claves de cada uno de los criptosistemas.
Denotaremos el cuerpo que utilizaremos como base como $\mathbb{K}$ y usaremos para la entrada y salida de variables sobre la función de clave pública las letras $\bar{x}$ e $\bar{y}$. Donde $\bar{x} \in \mathbb{K}^{n+1}$, que la escribiremos de la siguiente manera: $\bar{x} = (x_0,...,x_n)$, ocurrirá lo mismo con $\bar{y} = (y_0,...,y_n)$. Y definiremos la clave pública multivariante por una aplicación polinómica de $\mathbb{K}^{n+1}$ en $\mathbb{K}^{n+1}$.

$$\begin{array}{rccc} p\colon & \mathbb{K}^{n+1} & \longrightarrow & \mathbb{K}^{n+1} \\ & \bar{x} & \longmapsto & (p_0,...,p_n) \end{array}$$

Donde los $p_i(x) \in \mathbb{K}^{n+1}$ son polinomios. Esto se encontrará al alcance de todo el mundo, así pues cuando queramos codificar nuestro mensaje $\bar{x}$ será sustituida por el mensaje en el cuerpo que hayamos elegido y nuestros polinomios se evaluarán en esta $\bar{x}$ de manera que obtendremos así las $\bar{y}$. Ya hemos construido la base de nuestros criptosistemas del inverso del polinomio donde dicha función representa la clave pública.

3.2.1. Criptosistema MPS

El criptosistema MPS *Multivariant Polynomial Scheme*, llamado *esquema de polinomios multivariantes* es un tipo de criptosistema tal que como ya hemos definido su clave pública viene dada por una aplicación de un cuerpo de $n+1$ variables a otro cuerpo de $n+1$ variables; es decir:

$$\begin{array}{rccc} p\colon & \mathbb{K}^{n+1} & \longrightarrow & \mathbb{K}^{n+1} \\ & \bar{x} & \longmapsto & (p_0,...,p_n) \end{array}$$

Imaginemos que nosotros somos los dueños de esta clave pública, que alguien evalúa su mensaje en esos polinomios y que recibimos un mensaje codificado $\bar{y} = (y_0,...,y_n)$, supuestamente nosotros para poder recuperar el mensaje inicial lo que nos dice este método que deberíamos de realizar sería lo siguiente:

$$\begin{cases} y_0 = x_0 \\ y_1 = p_1(x_0) + x_1 \\ y_2 = p_2(x_0,x_1) + x_2. \\ . \\ . \\ . \\ y_n = p_n(x_0,...,x_n) \end{cases}$$

Es decir, lo que buscaríamos en realidad sería encontrar una solución a este sistema triangular, pero aquellos terceros que quisieran descubrir cual era el mensaje encriptado lo tendrían demasiado fácil, luego debemos buscar una forma de rebuscar más el poder encontrar la solución.

Lo que realizaremos para dificultar a terceros llegar a la solución consiste en que el vector $\bar{x} = (x_0,...,x_n)$ se convierta en un montón de combinaciones lineales de manera que ocultemos todavía más el mensaje, es decir, lo que realizamos son transformaciones a derecha e izquierda de dicho vector, que esto son transformaciones de filas y columnas, de manera que nuestro $\bar{x}$ se convierta en $\overline{x} = (\overline{x_0},...,\overline{x_n})$ y así cuando nosotros queramos decodificar el sistema tengamos que resolver lo siguiente:

$$\begin{cases} y_0 = \overline{x_0} \\ y_1 = p_1(\overline{x_0}) + \overline{x_1} \\ y_2 = p_2(\overline{x_0},\overline{x_1}) + \overline{x_2}. \\ . \\ . \\ . \\ y_n = p_n(\overline{x_0},...,\overline{x_n}) \end{cases}$$

Y aquí es donde nosotros por medio de la utilización de las bases de Groebner intentaremos llegar al cero de este sistema para así poder obtener el mensaje encriptado que nos querían hacer llegar.

3.2.2. Criptosistema MPK

El llamado *criptosistema multivariante de clave pública* o como es conocido, *Multivariant Public Key*, es un criptosistema que posee una clave pública dada por la siguiente aplicación:

$$\begin{array}{rccc} q\colon & \mathbb{F}_p^n & \longrightarrow & \mathbb{F}_p^n \\ & \bar{x} & \longmapsto & (p_0,...,p_{n-1}) \end{array}$$

Donde n es el número de variables que poseemos y p el número de elementos del cuerpo, luego nuestra pregunta será cómo debemos generar dichos polinomios $p_i \ \forall i \in \{0,...,n-1\}$. Veámoslo.

Así el método MPK se basa en la utilización de la exponenciación sobre $\mathbb{E}$, donde $\mathbb{E}$ es la extensión de grado n sobre el cuerpo $\mathbb{F}_p$.

$$\begin{array}{rccl} \varphi: & \mathbb{F}_p^n & \longrightarrow & \mathbb{E} \\ & x & \longmapsto & \varphi(x) = \chi = \sum_{i=0}^{n} x_i a^i \end{array}$$

Donde el vector $(1,...,a^n)$ formará la base de la extensión $\mathbb{E}$, así ahora debemos elegir un exponente $\theta \in \mathbb{Z}$ donde tomaremos $1+p^\theta$ tal que sea primo con p^n-1, puesto que así existirá nuestra clave privada que además al ser fijada de esta forma nos permitirá su inversión es decir poseeremos un e tal que igual que en el método RSA y gracias al uso del pequeño teorema de Fermat:

$$(\chi^{(1+p^\theta)})^e = \chi^{(1+p^\theta)e} = \chi^{k(p^n-1)}\chi = \chi$$

Ahora, elegiremos nosotros una transformación que en verdad será una aplicación de $\mathbb{E}$ en $\mathbb{E}$ en la que obtendremos una función cuadrática f de manera random que poseerá este estilo para así dificultar la desencriptación:

$$f = \sum_{0\le i\le j\le n, p^i+p^j<D} a_{i,j} x^{p^i+p^j} + \sum_{0\le k\le n, p^k<D} b_k x^{p^k} + c$$

A esta f la evaluaremos en $\chi^{(1+p^\theta)}$.

$$\begin{array}{rccl} f: & \mathbb{E} & \longrightarrow & \mathbb{E} \\ & \chi^{(1+p^\theta)} & \longmapsto & f(\chi(1+p^\theta)) \end{array}$$

Así y para concluir como nosotros queremos polinomios sobre las variables $(x_0,...,x_{n-1})$, lo que realizamos es otra vez el cambio de bases que pasa de la extension $\mathbb{E}$ a $\mathbb{K}^n$, es decir aplicamos φ^{-1} y así cada uno de los polinomios que son la clave pública vendrán dados por los coeficientes que se encuentran multiplicando a la base $\mathbb{E}$.

Luego recopilando lo que estamos realizando:

$$\begin{array}{ccc} \mathbb{K}^n & \xrightarrow{q} & \mathbb{K}^n \\ \downarrow{\scriptstyle \varphi} & & \uparrow{\scriptstyle \varphi^{-1}} \\ \mathbb{E} & \xrightarrow{f} & \mathbb{E} \end{array}$$

Luego ya podemos realizar para cualquier persona su clave pública.
Ahora veamos la codificación de un mensaje, lo que realizamos es primero elegir nuestro mensaje $\bar{x}$, después tomamos la clave pública de la persona a la que queremos enviarle nuestro texto en claro codificado, esto significa tomar los $p_i(x) \ \ \forall i \in \{0,...,n-1\}$ y evaluarlos en $\bar{x}$ y ya poseemos nuestro mensaje codificado donde ahora, el propietario de la clave secreta tiene la habilidad de poder encontrar la solución a este sistema, puesto que el conoce cómo se han obtenido los p_i:

$$\begin{cases} y_0 = p_0(x_0,...,x_{n-1}) \\ y_1 = p_1(x_0,...,x_{n-1}) \\ . \\ . \\ . \\ y_{n-1} = p_{n-1}(x_0,...,x_{n-1}) \end{cases}$$

Para poder hallarla teóricamente, el propietario ha de utilizar la transformación que hemos realizado mediante la f y el inverso de $(1+p^{\theta})$ para invertir cada una de las componentes de la clave pública. En verdad, para resolver este sistema usaremos las bases de Groebner, ya que el mensaje consiste en encontrar los ceros de este sistema.

Capítulo 4

Ejemplos de rotura del método de Polly Cracker. Un análisis cuantitativo.

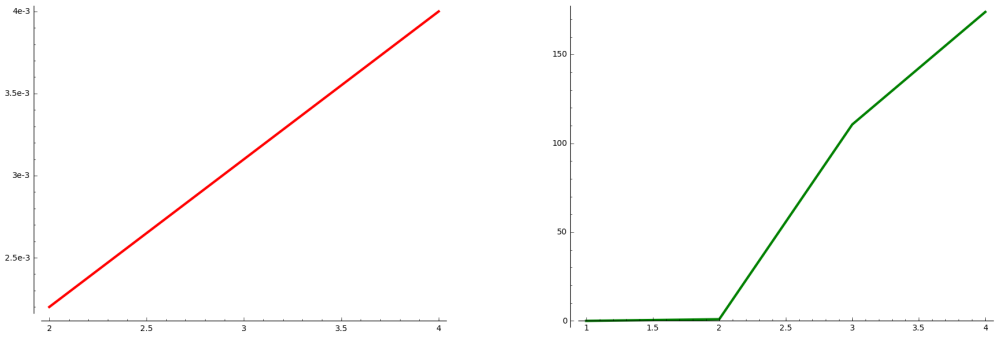
En este capítulo implementaremos un algoritmo en Sage, que se podrá ver en los anexos, para calcular el tiempo, en segundos, que tarda nuestro programa en romper por medio de bases de Groebner los criptosistemas de Polly Cracker.

Como habíamos dicho para poder codificar un mensaje con este método debíamos elegir una serie de variables como son:

- θ una variable que nos serviría para ocultar todavía mas el mensaje
- p el número de elementos del cuerpo $\mathbb{K}$
- n el grado de la extensión
- d que será el grado en el que queremos truncar los polinomios que obtengamos

Ya sabemos también que lo más usual es tomar $p \in \{2,3\}$ de manera que hemos variado esta constante en esos dos números y hemos observado qué es lo que ocurre. Advertir que Sage está programado para que detenga todo proceso que supere los 30 minutos de ejecución. Por lo tanto, cuando nuestros cálculos duren más de ese tiempo no obtendremos respuesta, aunque podemos asegurar que el coste computacional es mayor que 30 minutos. Para obtener estos valores de tiempo t, dada una lista n, p, θ, d hemos calculado la media del tiempo de ejecución del programa para 100 casos aleatorios.

Comencemos viendo la variación de t con respecto de d cuando $\theta = 3$ y $p = 2$:



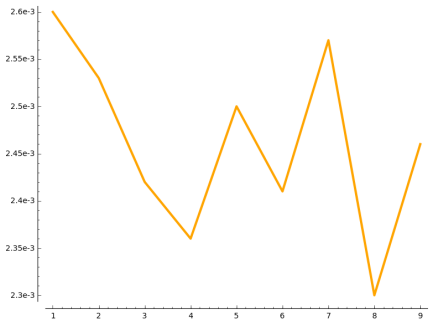
(a) $n = 2$ (b) $n = 3$

Figura 4.1: $\theta = 3$ y $p = 2$

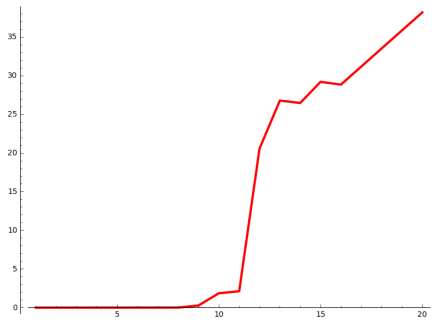
Podemos observar como con $n = 2$ existe un crecimiento proporcionado puesto que vemos que la gráfica de valores es prácticamente una recta con pendiente constante, mientras que con $n = 3$, cuando

$d = 3$ el coste computacional crece muy rápido y si $d = 5$ o superiores, Sage no nos proporciona ningún valor del tiempo ya que este sobrepasa la media hora. Tener en cuenta también que con $n = 4$ y $d = 1$ el tiempo que obtenemos es 0.24 segundos y con $n = 5$ y $d = 1$ el tiempo es 0.662 segundos pero con d's mayores no obtendremos respuesta.

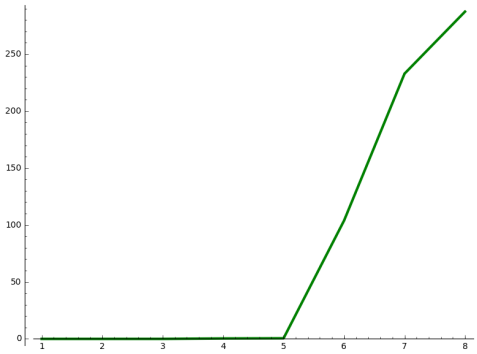
Veamos qué ocurre cuando $\theta = 3$ y $p = 3$:



(a) $n = 2$



(b) $n = 3$



(a) $n = 4$

Figura 4.3: $\theta = 3$ y $p = 3$

En estas gráficas podemos observar cómo en $n = 2$ el tiempo crece de manera muy lenta, además de tener la necesidad de destacar que romper este criptosistema es muy poco costoso computacionalmente puesto que no sobrepasa las milesimas de segundo.

En $n = 3$ vemos que hasta que no truncamos el grado del polinomio en 10, es decir, $d = 10$, no conseguimos que el proceso dure más de un segundo, esta gráfica para poder observarla mejor hemos hecho que llegue hasta $d = 20$, sin embargo el verdadero crecimiento se inicia a partir de $d = 20$, donde cuando tomamos $d = 23$ el tiempo que tarda en darnos solución es aproximadamente 20 minutos. También cabe destacar que con $p = 3$ el coste es mucho menor cuando vamos cambiando la n que con $p = 2$.

También podemos observar que al ir aumentando la n necesitamos una d menor para que el coste computacional aumente de manera rápida, como podemos ver en $n = 4$, añadir que para $n = 5$ o superiores con d's muy pequeñas nuestro programa ya no nos saca por pantalla el tiempo que le cuesta.

Dibujemos como sería la superficie en la que fijando la p y la θ variamos la n, la d y el tiempo en segundos que le cuesta, añadir que en aquellos lugares donde nosotros no damos valores del tiempo puesto que no los podemos conseguir o porque no podemos truncar nuestro polinomio a más de lo que ya lo hemos hecho por defecto Sage lo pintará como 0. Podemos observar en dicha superficie o también en los gráficos que en números pares por lo general o la función crece o por lo general el coste

computacional es menor que en el d impar anterior.

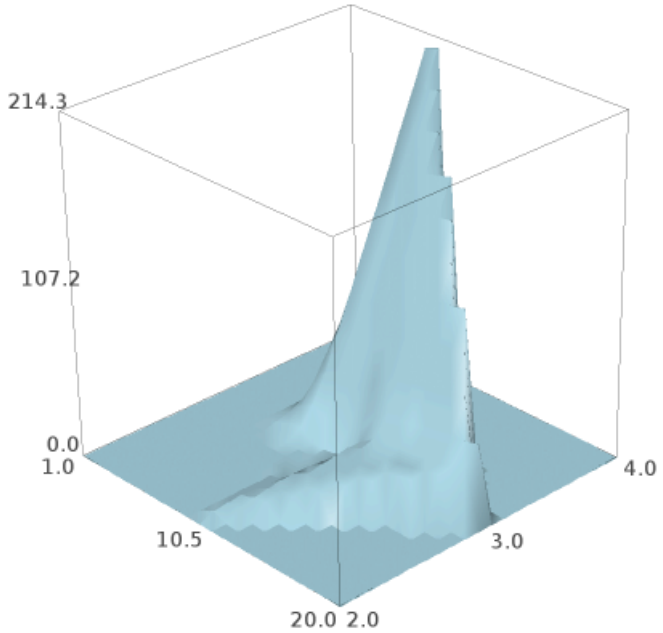
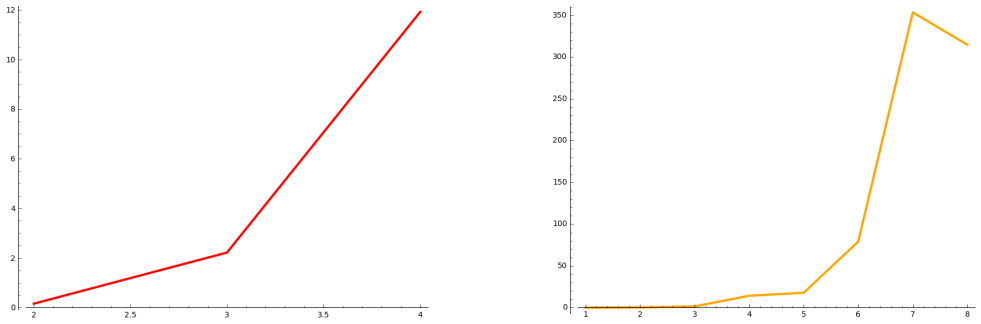


Figura 4.4: Superficie con $\theta = 3$ y $p = 3$

Tomemos ahora $\theta = 7$ y volvamos a ver cual es su coste computacional, para esta nueva constante que tan solo dista de 4 unidades de la que hemos tomado en la primera prueba el tiempo del cpu es enorme según vamos aumentando la n de tal forma que solo podemos obtener valores significativos para $p = 2$ o $p = 3$ cuando $n = 2$. Veamos estas gráficas.



(a) $p = 2$ (b) $p = 3$

Figura 4.5: Realizamos cambio de p

Como podemos observar e igual que nos ha pasado antes con $\theta = 3$ el coste computacional para d's pequeñas es menor con $p = 3$ que con $p = 2$ pero en el momento en que esta variable, d, va aumentando se va incrementando de forma significativa, tanto que en valores tan pequeños como puede ser $n = 3$ y $d = 2$ ya no nos da valor puesto que tarda más de media hora.

Así pues podemos deducir que para valores muy pequeños de n, d y θ los costes computacionales ya empiezan a ser grandes e incluso de más de 30 minutos en algunos casos, luego según se vayan

aumentando los valores los tiempos de rotura de este criptosistema pueden llegar a ser garrafales y por lo tanto podemos deducir que es un método bastante seguro, puesto que seguramente deberíamos tener a nuestro ordenador trabajando durante una cantidad ingente de tiempo para poder obtener una respuesta.

Bibliografía

[1] DAVID COX, JOHN LITTLE, DONAL O'SHEA, *Ideals, Varieties and Algorithms*, Colección Springer, Tercera Edición, 2007.

[2] MASSIMILIANO SALA, TEO MORA, LUDOVIC PERRET, SHOJIRO SAKATA, CARLO TRAVERSO, *Gröbner Bases, Coding and Cryptography*, Colección Springer, 2009.

[3] DURÁN, HERNÁNDEZ, MUÑOZ, *El criptosistema RSA*, Editorial RA-MA, 2005.

[4] PAAR-PELZL, *Understanding Cryptography*, Colección Springer, 2010.

Índice alfabético

Anexo

Durante los capítulos previos hemos enunciado y hecho uso en alguna demostración del teorema de Hilbert y del lema de Dickson asumiendo que eran ciertos ya que solamente necesitabamos el resultado, así pues veamos ahora las demostraciones de dichos enunciados.

Teorema. ***Teorema de la base de Hilbert***

1. *Si I es un ideal de R entonces está finitamente generado.*
2. *(Condición de cadena ascentente) Si $I_1 \subseteq I_2 \subseteq ... \subseteq I_n \subseteq ...$ es una cadena ascendente de ideales en $\mathbb{K}[x_1,...,x_n]$ entonces existe un N tal que $I_N = I_M$ para todo $N \leq M$.*

Demostración. Realizaremos la demostración por partes comenzaremos viendo que si I es un ideal entonces está finitamente generado. Si $I = \{0\}$, tomaremos el conjunto generado como $\{0\}$, que trivialmente es finito.
Si I contiene algún polinomio que sea distinto de cero. Entonces nuestro conjunto generador $\{g_1,...g_n\}$ para I puede ser construido de la siguiente forma.

Nosotros sabemos que existe $\{g_1,...,g_n\} \in I$ tal que $\langle lt(I)\rangle = \langle lt(g_1),...,lt(g_n)\rangle$ lo que nos dice que $I = \langle g_1,...,g_n\rangle$.
Es claro que $\langle g_1,...,g_n\rangle \subset I$ ya que $g_i \in I$. Asú pues, veamos el otro contenido, sea $f \in I$ un polinomio si aplicamos el algoritmo de la división para dividir f entre $\langle g_1,...,g_n\rangle$ entonces obtenemos la siguiente expresión:

$$f = a_1g_1 + ... + a_ng_n + r$$

donde ningún termino de r es divisible por los $lt(g_i)$ $\forall i \in \{1,...,n\}$, lo que implica que $r = 0$. Veámoslo detalladamente, sabemos que:

$$r = f - a_1g_1 + ... + a_ng_n.$$

Si $r \neq 0$ implicaría que $lt(r) = \langle lt(I)\rangle = \langle lt(g_1),...,lt(g_n)\rangle$, pero nosotros sabemos que $lt(r)$ no puede ser divisible por ningún $lt(g_i)$. Luego $r = 0$ y conseguimos la primera parte del teorema.

Vayamos con la segunda parte del teorema, sea dado $I_1 \subset I_2 \subset I_3 \subset ...$, consideramos $I = \cup_{i=1}^{\infty} I_i$. Empezaremos viendo que I también es un ideal en $\mathbb{K}[x_1,...,x_n]$.

- Como $0 \in I_i$ $\forall i$ entonces $0 \in I$
- Sea $f \in I_i$, $g \in I_j$ para algunos i y j. Sin embargo, como los I_i forman una cadena ascendente podemos renombrarlos de manera que $i \leq j$ lo que implicará que $f, g \in I_j$ y como consecuencia $f + g \in I_j$ y por lo tanto $f + g \in I$.
- De forma muy parecida, sea $f \in I_i$ para algún i y $r \in \mathbb{K}[x_1,...,x_n]$ y $rf_i \in I$.

Por lo tanto I es un ideal. Por la demostración previa que hemos realizado sabemos que I debe estar finitamente generado luego, $I = \langle f_1,...,f_n\rangle$. Pero cada uno de los generadores está contenido en uno de los $I_{j,i}$, de tal forma que $f_i \in I_{j,i}$ $\forall j$.

Tomamos pues un N que sea el máximo de los j,i. Entonces por definición de cadena ascendente $f_i \in I_N$ $\forall i$.

$$I = \langle f_1, ..., f_n \rangle \subset I_N \subset I_{N+1} \subset .... \subset I.$$

Esto por cadena ascendente podemos afirmar que se estabilizará en un N y todos los I_M tal que $N \leq M$ serán iguales a I_N. □

Lema 1. ***Lema de Dickson.*** *Sea $I = \langle x^\alpha : \alpha \in A \rangle \subset \mathbb{K}[x_1, ..., x_n]$ un ideal monomial . Entonces I puede ser escrito de la siguiente forma $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$ donde los $\alpha(i) \in A$. En particular, I tiene una base finita.*

Demostración. Procedamos por inducción.
Sea $n = 1$ esto implica que I es generado por monomios x_1^α donde $\alpha \in \mathbb{Z}_{\leq 0}$.
Sea ahora $\beta \leq \alpha$ el menor elemento que existe en $A \subset \mathbb{Z}_{\leq 0}$ $\forall \alpha \in A$, luego x_1^β divide a todos los x_1^α, esto quiere decir que $I = \langle x_1^\beta \rangle$.

Luego lo hemos demostrado para $n = 1$ veamos que ocurra para $n > 1$, por lo tanto asumámoslo cierto para $n > 1$ hasta $n - 1$ y probémoslo para n. Luego tomamos las variables $\{x_1, ..., x_{n-1}, y\}$ que serán monomios en $\mathbb{K}[x_1, ..., x_{n-1}, y]$ que pueden ser escritos como $x^\alpha y^m$ donde $\alpha = (\alpha_1, ..., \alpha_{n-1}) \in \mathbb{Z}_{\leq 0}^{n-1}$ y $m \in \mathbb{Z}_{\leq 0}$.

Supongamos que $I \subset \mathbb{K}[x_1, ..., x_{n-1}, y]$ es un ideal monomial. Busquemos pues los generadores de I, para ello, sea J un ideal de $\mathbb{K}[x_1, ..., x_{n-1}]$ generado por los monomios x^α para cada $x^\alpha y^m \in I$ para algún $m \geq 0$.
Como J es ideal monomial de $\mathbb{K}[x_1, ..., x_{n-1}]$ por hipótesis de inducción sabemos que las x^α generan finitamente a J, es decir, $J = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$.
Para cada $i \in \{1, ..., s\}$ la construcción de J nos dice que $x^{\alpha(i)} y^{m_i} \in I$ para algún $m_i \geq 0$. Tomemos como m al mayor de todos los m_i y entonces para cada $k \in \{1, ..., m-1\}$, consideramos el ideal $J_k \subset \mathbb{K}[x_1, ..., x_n - 1]$ generado por los monomios x^β tal que $x^\beta y^k \in I$. Utilizando la hipótesis de inducción, J_k tiene un conjunto de monomios generadores finito, $J_k = \langle x^{\alpha_k(1)}, ..., x^{\alpha_k(s)} \rangle$.

Luego podemos asegurar que I está generado por los monomios de la siguiente lista:

$$\begin{gathered} \text{Para } J : x^{\alpha(1)} y^m, ..., x^{\alpha(s)} y^m \\ \text{Para } J_0 : x^{\alpha_0(1)}, ..., x^{\alpha_0(s)} \\ \text{Para } J_1 : x^{\alpha_1(1)} y, ..., x^{\alpha_1(s)} y \\ ... \\ \text{Para } J_{m-1} : x^{\alpha_{m-1}(1)} y^{m-1}, ..., x^{\alpha_{m-1}(s)} y^{m-1} \end{gathered}$$

Notemos que cada monomio de I es divisible por uno de la lista, probémoslo. Sea $x^\alpha y^p \in I$.

- Si $p \geq m$ esto implica que $x^\alpha y^p$ es divisible entre $x^{\alpha(i)} y^m$ por construcción de J.
- En el caso opuesto que $p \leq m$ esto implica que $x^\alpha y^p$ es divisible por algún $x^{\alpha_p(j)} y^m$ por propia construcción de J_p.

Para completar la demostración, necesitamos ver que el conjunto finito de generadores puede ser elegido desde un conjunto dado de generadores de un ideal. Si tomamos las variables $\{x_1, ..., x_n\}$ nuestro ideal monomial será $I = \langle x^\alpha; \alpha \in A \rangle \subset \mathbb{K}[x_1, ..., x_n]$. Luego necesitamos ver que I está generado finitamente por los x^α con $\alpha \in A$.
Pero sabemos que $I = \langle x^{\beta(1)}, ..., x^{\beta(s)} \rangle$ para algún monomio $x^{\beta(i)}$ en I, como $x^{\beta(i)} \in I = \langle x^\alpha; \alpha \in A \rangle$ esto implica que $x^{\beta(i)}$ es divisible por $x^{\alpha(i)}$ para algún $\alpha(i) \in A$. Y por lo tanto podemos fácilmente ver que $I = \langle x^{\alpha(1)}, ..., x^{\alpha(s)} \rangle$.

□

Por último, en el capítulo 4 hemos obtenido una serie de resultados gráficos gracias a un programa implementado en Sage, su código era el siguiente:

```
reset()
def truncar(poli,grado):
    return sum([i[0]*x^i[1] for i in poli.coefficients() if i[1]<=grado])

#p caracteristica
p=2
#grado de la extension
n=3
#d grado máximo de polinomios
d=4

F=GF(p)
E=GF(p^n,'a')
k.<a> = GF(p^n, repr='int')
xs1=list(var('x%d' %i) for i in range(n))
R = PolynomialRing(k,names=xs1,order="lex")
ax=[(xs1[i]*a^i) for i in range(n)]
suma=sum([ax[i] for i in range(n)])

tiempos=[]
for h in range(100):
    A=random_matrix(ZZ,n+1,n+1,x=d+1)
    C=matrix(ZZ,[j*[0]+[A[j][i+j] for i in range(n-j+1)] for j
    in range(n+1)])
    X=vector([1]+[x^p^(i) for i in range (n)])
    P=X*C*X
    P=P.expand()
    Pt=truncar(P,d)
    f1=R(Pt(x=(suma)^(1+2^3)))
    v=[[(k.vector_space()).coordinates(k(f1.monomial_coefficient(R(i)))),
    i] for i in f1.monomials() if i<>1]
    I=[sum([i[0][j]*i[1] for i in v]) for j in range(n)]*R
    w=cputime(t=0)
    A=I.groebner_basis()
    tiempos=tiempos+[cputime(w)]

mean(tiempos)
```

Naturalmente, tanto p, como d, como n son variables y es lo que hemos ido cambiando según los datos que queríamos analizar. También en $f1$ hemos modificado ese 3 que es el que nos hace la función de θ y en cuanto al rango de tiempos, como siempre queríamos realizar 100 pruebas, cuando veíamos que era muy costoso calcularlo en una de una sola vez, lo íbamos variando aunque siempre el número de datos a analizar que obteníamos era 100.