UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

Métodos algebraicos en criptografía multivariable

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Jorge Linde Díaz

Director

Ignacio Luengo Velasco

Madrid
Ed. electrónica 2019

Métodos Algebraicos en Criptografía Multivariable

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Jorge Linde Díaz

DIRECTOR
Ignacio Luengo Velasco

Madrid, 2018

A Juan Antonio Martín Rollan

Agradecimientos.

A mi novia Ana, por ayudarme con el desarrollo de la tesis y por su confianza y ánimo durante todos estos años.

A mis padres Pepe y Sarah, por apoyarme y animarme a estudiar siempre; además, a mi padre por su esfuerzo para ayudarme con el desarrollo de la tesis.

A Sheriff, por el apoyo mutuo durante el doctorado y por la motivación que me ha dado.

A grandes profesores que he tenido durante la carrera como: María Emilia Alonso, Javier Lafuente, María Jesús de la Puente, María Concepción Fuertes, Enrique Arrondo, Francisco Javier Gallego y en especial a José Manuel Gamboa, Daniel Azagra y Vicente Muñoz.

A mi director Ignacio Luengo Velasco.

Al centro ICMAT por financiar mi doctorado con el contrato SVP-2014-068440 y al Departamento de Álgebra de la Facultad de Ciencias Matemáticas de la Universidad Complutense de Madrid.

Índice general

Resumen

- Título de la tesis: Métodos Algebraicos en Criptografía Multivariable.
- Introducción y motivación: "La criptografía de curva elíptica no es la solución a largo plazo que esperábamos que fuera. Por tanto, nos hemos visto obligados a replantear nuestra estrategia". Con estas palabras alertaba en el 2015 la Agencia Nacional de Seguridad estadounidense (NSA) sobre la necesidad de encontrar nuevas herramientas criptográficas que permitan mantener la seguridad en Internet. En un comunicado de su web, la agencia ha mostrado su interés en "iniciar una transición a algoritmos potencialmente resistentes a ordenadores cuánticos en un futuro no muy lejano". Consideran un objetivo principal disponer de herramientas de seguridad ante un posible ordenador cuántico.

  En las últimas décadas Internet ha adquirido un papel central en la sociedad. Uno de los pilares sobre los que se sostiene la red es la criptografía de clave pública, que permite las comunicaciones privadas, la identificación de los usuarios en un servidor y la firma de documentos. Entre los diferentes métodos que se utilizan para llevar a cabo este tipo de algoritmos está el algoritmo RSA, que se basa en la factorización de números enteros muy grandes. Sin embargo, en el año 1994 Peter Shor presentó un algoritmo que permite factorizar números enteros grandes y podría echar abajo este sistema. La única razón por la que aún el algoritmo de Shor no ha puesto en peligro la seguridad de Internet es que para ello necesita un ordenador cuántico de miles de q-bits. Por el momento dicho ordenador no existe, pero los progresos actuales en computación cuántica permiten pensar que dentro de unas décadas se podrá construir un ordenador cuántico con esas características. Por tanto, es necesario enfrentar este nuevo escenario lo antes posible.

  En el congreso de criptografía post-cuántica, PQCrypto 2016, el Instituto Nacional de Estándares y Tecnología Americano (NIST) ha lanzado una llamada de atención sobre la urgencia de encontrar y estandarizar en los próximos años un sistema crip-

tográfico que sea resistente al Algoritmo de Shor. La criptografía post-cuántica es el campo de estudio de este tipo de sistemas criptográficos que no son afectados por los ordenadores cuánticos. Una de las técnicas empleadas es la criptografía multivariable, en ella se estudian los sistemas cuya dificultad radica en la complejidad de resolver sistemas de ecuaciones polinomiales en muchas variables. Este trabajo pretende ayudar a construir un sistema multivariable que pueda llegar a ser uno de los seleccionados por el NIST para su estandarización.

- Síntesis: los objetivos principales de esta tesis han sido dos. El primero, mostrar las propiedades de las sucesiones semi-regulares y del índice de regularidad para entender mejor el comportamiento del algoritmo F4, útil para el estudio de los sistemas MPKC. El segundo, estudiar el sistema ME propuesto por Ignacio Luengo, un tipo de sistema MPKC englobado en la criptografía post-cuántica con la intención de patentarlo y llevarlo al NIST para su estandarización. Dichos objetivos están motivados por la necesidad de encontrar un nuevo tipo de criptografía como se muestra en el capítulo 1, en este capítulo también mencionamos las ventajas e inconvenientes de la criptografía multivariable dentro de la criptografía post-cuántica.

  Para el primer objetivo, hemos mostramos en el capítulo 1 los objetos y la teoría matemática necesaria para su desarrollo. Por otra parte, en el capítulo 2 presentamos ciertos algoritmos que se usan para el estudio de los sistemas MPKC, y en particular, el F4. Dichos algoritmos atacan la estructura algebraica del sistema para poder hallar una solución y así poder obtener el texto original de un texto cifrado. En el capítulo 3 es donde nos centramos en el estudio de las sucesiones semi-regulares y del índice de regularidad. Con dicho estudio somos capaces de entender mejor las debilidades de los sistemas MPKC. En este capítulo hemos conseguido resultados novedosos, por ejemplo hemos mostrado la relación que hay entre las dos definiciones distintas de sucesión semi-regular que se encuentran en la literatura, gracias a la proposición 3.18:

  Proposición. *Sea* $f_i \in A^{i-1} = S/I^{i-1}$ *una forma . Se verifica que la aplicación multiplicación por* f_i *es suprayectiva para el grado a si, y solo si,* $\dim A_a^i = 0$.

  mostramos que la de definición de Faugère es más débil. También demostramos un teorema fundamental de las sucesiones semi-regulares según la definición de Faugère, el teorema 3.28:

  Teorema. *Sea* $I = (f_1, \ldots, f_m)$ *un ideal homogéneo con* $\deg f_i = d_i$*, entonces* $(f_1, \ldots, f_m)$ *forman una sucesión semi-regular si, y solo si, la serie de Hilbert viene*

dada por:

$$HS_{(f_1,\ldots,f_s)} = \sum_{k=1}^{\infty} c_k z^k = \left[\frac{\prod_{i=1}^{s}(1-z^{d_i})}{(1-z)^n}\right]$$

Gracias a él somos capaces de establecer varias conclusiones, como la observación 3.38, podemos estimar el comportamiento del algoritmo F4 para sucesiones semi-regulares según Faugère y obtenemos ciertas gráficas que nos muestran cómo se comporta el d_{reg} para ideales formados por este tipo de sucesiones. En el capítulo 4 volvemos a retomar los sistemas multivariables y mostramos cómo se comporta el d_{reg} para el sistema HFE. También presentamos una idea novedosa, el 'grado intrínseco' del sistema HFE, un tipo de grado con el que se puede entender mejor el comportamiento del F4 sobre el HFE.

Para el segundo objetivo lo primero que hacemos es presentar el sistema ME en el capítulo 5 que tiene la siguiente forma:

$$P = P_1 \circ Q_1 \circ P_0$$

donde P_0, P_1 son aplicaciones exponenciales y Q_1 es una aplicación 'triangular'. A diferencia del resto de sistemas MPKC este sistema tiene muy pocos monomios, gracias a la estructura interna de Q_1. Más tarde presentamos el método de descifrado del sistema, así como ciertas relaciones que verifican las ecuaciones que más tarde nos serán útiles para su análisis. Finalmente, analizamos el sistema gracias al método de las claves equivalentes que he desarrollado. Este método se lleva a cabo gracias a las observaciones 5.7, 5.8, 5.9, 5.10, 5.11. Con ellas nos damos cuenta de que podemos hacer suposiciones de nuestro sistema para luego poder atacarlo. El análisis del sistema es complejo y tiene varias etapas, la idea es ir obteniendo información de la clave privada gracias a las observaciones anteriores y las relaciones mencionadas previamente. Finalmente, nos damos cuenta de que el sistema está roto y no se puede patentar. En el capítulo 6 mostramos una variante del HFE que tampoco es segura, así como el sistema GTS que he diseñado y que funciona bien en los experimentos que he realizado. Por otra parte, este estudio ha ayudado a la elaboración del sistema DME, el cual ha sido patentado y está siendo revisado por el NIST.

Otro resultado de la tesis ha sido los programas que se muestran en el capítulo 7, en ellos ha sido implementado el sistema HFE y el sistema GTS, también se han desarrollado herramientas para el estudio del HFE, del índice de regularidad y de las sucesiones semi-regulares. Gracias a ellos he podido generar todas las gráficas mostradas en esta tesis.

- Conclusiones:
  - El sistema presentado ME no es seguro, y por tanto, no se puede patentar.
  - El método de las claves equivalentes desarrollado en esta tesis es una herramienta útil para el estudio de sistemas MPKC.
  - La definición de sucesión semi-regular dada por Faugère es más débil que la estándar.
  - Podemos caracterizar las sucesiones semi-regulares a partir de la Serie de Hilbert asociada.
  - El grado intrínseco de un sistema HFE permite caracterizar, mejor que el grado, la complejidad del sistema.
  - El sistema HFE + exponencial no es seguro.
  - No podemos concluir que el sistema GTS no sea seguro.

Abstract

- Thesis title: Algebraic Methods in Multivariate Cryptography
- Introduction and Motivation: 'Elliptic Curve Cryptography is not the long term solution many once hoped it would be. Thus, we have been obligated to update our strategy'. With these words, the National Security Agency (NSA) of the United States of America alerted about the necessity of finding new cryptographic tools that would allow us to maintain security over the Internet. In a statement on their website, the agency has shown interest in 'initiating a migration to potentially quantum-resistant cryptographic algorithms in the near future'. They consider essential the disposal of security tools against a possible quantum computer. One of the main corner stones on which the network is supported is public key cryptography (PKC), which allows private communications, authentication of users in a server and signature of data. Among the different PKC schemes, there is the RSA algorithm, whose security is dependent on the difficulty of factorizing a big composite number. However, in 1994 Peter Shor presented a polynomial-time factorization algorithm, which would make RSA insecure. The only reason why Shor's algorithm has not endangered Internet's security is that it was designed to be programmed in a quantum computer with thousands of q-bits. Currently, such a computer does not exist, but the ongoing progress in quantum computation suggests that a quantum computer with the necessary characteristics will be plausible within some decades. Therefore, it is necessary to confront such a new scenario as soon as possible. On the International conference on Post-Quantum Cryptography, PQCrypto2016, the American National Institute on Standards and Technology (NIST) has pointed the importance and urgency of finding and standardizing a cryptographic scheme that is resistant to Shor's algorithm in the coming years. Post-quantum cryptography is the field of research of such cryptographic schemes that are not affected by the existence of a machine able to reproduce Shor's algorithm. One of the lines of research of post-quantum crypto-

graphy is Multivariate Cryptography. This line studies the schemes whose security resides on the difficulty of solving polynomial systems with many variables. This thesis pretends to support the construction of a multivariate system that could be selected by the NIST for the standardization of post-quantum cryptography.

- Summary: The principal objectives of this thesis are twofold. On the one hand, show the properties of semi-regular sequences and the degree of regularity for a better understanding of the behaviour of the F4 algorithm, useful for the study of MPKC systems. On the other hand, we study the ME system proposed by Ignacio Luengo, a MPKC system encompassed in post-quantum cryptography with the intention of patenting it and taking it to the NIST for standardization. These objectives are motivated by the need of finding a new kind of cryptography as shown in chapter 1, where we also mention the advantages and inconveniences of multivariable cryptography within post-quantum cryptography.

  For the first objective, we begin by showing, in chapter 1, the objects and mathematical theory needed for its development. Secondly, chapter 2 presents certain algorithms used for the study of MPKC systems in general, and for the study of the F4 algorithm in particular. These algorithms attack the algebraic structure of the system in order to find a solution to the polynomials, and hence, decrypt the ciphertext to its original plaintext. Chapter 3 presents a study of semi-regular sequences and the degree of regularity which feeds us with a better understanding of the MPKC system's weaknesses. Moreover, in this chapter we present original results, proving the relation between two distinct definitions of semi-regular sequences that are found in the literature, by the proposition 3.18:

  Proposición. *Let $f_i \in A^{i-1} = S/I^{i-1}$ be a form. The application product by f_i is surjective for the degree a if, and only if,* $\dim A_a^i = 0$.

  We prove that Faugère's definition is weaker than normal definitoin. Furthermore, we prove a fundamental theorem of semi-regular successions following Faugère's definition, the theorem 3.28:

  Teorema. *Let $I = (f_1, \ldots, f_m)$ be an homogeneous idel with* $\deg f_i = d_i$*, them $(f_1, \ldots, f_m)$ form a semi-regular sequence if, and only if, Hilbert's Series is given by:*

$$HS_{(f_1,\ldots,f_s)} = \sum_{k=1}^{\infty} c_k z^k = \left[\frac{\prod_{i=1}^{s}(1-z^{d_i})}{(1-z)^n}\right]$$

Using this theorem we are able to draw several conclusions, such as observation 3.38, where we can estimate the behaviour of the F4 algorithm for semi-regular sequences following Faugère's definition and obtain graphs which allow us to study the behaviour of d_{reg} for ideals formed by this type of successions. In chapter 4, we retake multivariable systems to show the behaviour of d_{reg} for the HFE system. I also present a novel idea, named as the 'intrinsic degree' of the HFE system, a type of degree with which a better understanding the behaviour of the F4 over the HFE is attained.

For the second objective, our first step is to present the system, which has the following shape:

$$P = P_1 \circ Q_1 \circ P_0$$

where P_0, P_1 are exponential applications and Q_1 is a 'triangular' application. Contrarily to the other available MPKC systems, our proposal has a small number of monomials, thanks to the internal structure of Q_1. Later, we present the decryption method of the scheme, as well as several relations that verify the equations, which will later be of use for its analysis. Finally, we analyse the system thanks to the method of equivalent keys that I have developed. This method is carried out thanks to observations 5.7, 5.8, 5.9, 5.10 and 5.11. These observations motivate the assumptions we make of our system to later attack it. The analysis of the system is complex and divided in several stages. The idea is to obtain information of the private key thanks to the observations and relations previously mentioned. Our efforts results in breaking the system, showing that it is not secure and can, hence, not be patented. In chapter 6, we present a HFE variant that is insecure, as well as a the GTS system that I have designed and works well in the experiments. Moreover, this study has been of use for the elaboration of the DME system, which has been patented and is being revised by the NIST.

Another result of the thesis has been the programs shown in the chapter 7, in which the HFE and GTS systems have been implemented. Moreover, we have developed tools for the study of HFE, the degree of regularity and semi-regular sequences. Thanks to these, we have been able to generate all graphics presented in this thesis.

- Conclusions:
  - The presented scheme, ME, is not secure and, hence, cannot be patented.
  - The method of equivalent keys presented in this thesis is a useful tool for the study of MPKC systems.

- The definition of semi-regular sequences given by Faugère is weaker than the standard definition.
- We can characterize the semi-regular sequences from the associated Hilbert Series.
- The intrinsic degree of a system HFE enables us to characterize, better than the degree, the complexity of the system.
- The HFE system + exponential is not secure.
- We cannot draw conclusions on the insecurity of the GTS system.

Capítulo 1

Introducción

En las últimas décadas internet ha adquirido un papel central en la sociedad. Hoy en día internet es indispensable para nuestra sociedad ya que es necesario para las comunicaciones a distancia, la transmisión de información relevante, el comercio online, etc. Uno de los pilares sobre los que se sostiene la red es la criptografía que permite, por ejemplo, comunicaciones privadas, la identificación de los usuarios en un servidor y la firma de documentos, entre otras cosas.

La historia de la criptografía puede dividirse en tres eras: la primera es la criptografía clásica o criptografía simétrica, con sistemas como la Cifra César. Dichos sistemas eran rápidos, pero la mayoría se podían resolver eficientemente incluso sin un ordenador. Después de años de investigación en criptografía simétrica aparecieron los ordenadores, de hecho una de las iniciativas del desarrollo de los ordenadores fue la necesidad de romper el sistema de cifrado de la Máquina Enigma. Con los ordenadores gran parte de los sistemas simétricos podían romperse fácilmente, lo que llevó a un nuevo tipo de criptografía, la criptografía de clave pública o asimétrica. La criptografía de clave pública se basa en la existencia de problemas matemáticos que son muy difíciles de resolver. Esta nueva era de la criptografía empezó con el famoso trabajo de Diffie y Hellman, en el que se desarrolla el problema del logaritmo discreto DLP [DH06], el cual se utiliza en el sistema de intercambio de claves de Diffie y Hellman. Posteriormente Rivest, Shamir y Adleman propusieron el RSA [RSA78], uno de los criptosistemas de intercambio de claves más famoso y más seguro utilizados hoy en día basado en la factorización de números compuestos. Actualmente se utilizan sistemas mixtos de clave pública y clave privada para la comunicación por internet. Se podría decir que actualmente se está entrando en la tercera era de la criptografía, ya que en 1994 Peter Shor desarrolló un algoritmo cuántico [Sho97], el cual puede romper sistemas basados en la

factorización y el DLP en un tiempo polinomial con un ordenador cuántico suficientemente grande. Se estima que 'suficientemente grande' sería un ordenador con unos miles de qubits. Dicho algoritmo rompería sistemas como por RSA, DSA (algoritmo de firma digital basado en el logaritmo discreto) y ECDSA (algoritmo de firma digital sobre curvas elípticas). Aún no se ha conseguido construir un ordenador cuántico lo suficientemente potente como para romper dichos criptosistemas, aunque se estima que en unas pocas décadas se pueda llegar a construir un ordenador con dichas características. Por lo tanto, aunque hoy en día dichos sistemas se consideran seguros se tienen que buscar alternativas para un futuro cercano.

De hecho, una de las principales preocupaciones a día de hoy en este sector es lo que se conoce como *la seguridad hacia atrás en el tiempo*. Esto hace referencia al hecho de que todo lo que se está cifrando hoy en día podrá ser descifrado cuando se construya un ordenador cuántico (lo suficientemente potente). Por tanto, se debe hacer el cambio de la criptografía de clave pública a la criptografía post-cuántica unos cuantos años antes de que estos ordenadores aparezcan, sino toda la información sensible de los años previos a la aparición de los ordenadores cuánticos podrá ser descifrada. Por ello hoy en día se está concienciando a los investigadores de la importancia de desarrollar sistemas post-cuánticos seguros y eficientes. En el congreso mundial sobre criptografía post-cuántica del año 2016 celebrado en Japón, el instituto Nacional de Estándares y Tecnología Americano (NIST) ha lanzado una llamada de atención sobre la urgencia de encontrar y estandarizar en los próximos años un sistema criptográfico que sea resistente al algoritmo de Shor. De manera que internet siga siendo como se conoce hoy en día, incluso cuando aparezcan los ordenadores cuánticos. Actualmente (2018) el NIST está llevando a cabo la investigación de cual de los sistemas presentados será el próximo estándar post-cuántico. Entre los estudiados está el sistema DME (Double Matrix Exponentiation) [Lue18], desarrollado por Ignacio Luengo y para su desarrollo ha sido muy útil el criptoanálisis del sistema ME descrito en esta tesis.

En esta tercera era, una de las áreas actuales de investigación es la criptografía post-cuántica, la cual se encarga de desarrollar sistemas criptográficos que se puedan ejecutar en un ordenador convencional y que, además, sean potencialmente resistentes a ataques con ordenadores cuánticos. Aún no se ha conseguido encontrar un sistema de este tipo que sea eficiente, seguro y fiable. Por lo tanto, aún no estamos preparados para hacer el cambio a la criptografía post-cuántica.

Criptosistema	Tipo	Propósito	Situación futura
AES-256	Simétrico	Encriptación	Necesitará duplicar los tamaños de la entrada
SHA-256		Función Hash	Necesitará un aumento de los tamaños de salida
RSA	Asimétrico	Firmas e intercambio de claves	No será seguro
ECDSA, ECDH (Criptografía sobre Curvas Elípticas)	Asimétrico	Firmas e intercambio de claves	No será seguro
DSA (Criptografía sobre Cuerpos Finitos)	Asimétrico	Firmas	No será seguro

Cuadro 1.1: Impacto de la computación cuántica en la criptografía actual

1.1. Criptografía Post-Cuántica

En los últimos años se han desarrollado algoritmos cuánticos exponencialmente más eficientes, en la resolución de ciertos problemas, que los algoritmos clásicos. El número de problemas que admiten una mejora exponencial son reducidos lo que a día de hoy, asegura que existen problemas que actualmente son difíciles de resolver incluso para un ordenador cuántico. También es cierto que algunos algoritmos se han podido mejorar aunque no de una manera exponencial. Lo cual hace que los parámetros en ciertos sistemas criptográficos sean reconsiderados. Con la aparición de los ordenadores cuánticos, el panorama de los sistemas criptográficos más relevantes hoy quedará como se muestra en la tabla 1.1

Como se ve, con la llegada de los ordenadores cuánticos los protocolos de firma y establecimiento de claves no serán seguros. También es cierto que, aunque el algoritmo de búsqueda exhaustiva de Grover [Gro96] mejora de forma cuadrática los algoritmos clásicos, esto no compromete, a priori, la seguridad de los sistemas de encriptación simétricos, lo único que se tendrá que hacer será duplicar el tamaño de la clave para obtener el mismo nivel de seguridad. Además, como se muestra en el artículo [BBBV96], no se puede desarrollar un algoritmo de búsqueda exhaustiva que sea exponencialmente más rápido que los actuales, lo cual lleva a pensar que los sistemas de cifrado simétricos no quedarán comprometidos por los ordenadores cuánticos.

En esta sección resumiré brevemente qué es un ordenador cuántico y en qué situación se encuentran a fecha de hoy dichos ordenadores, para más detalles [NC10]. Cada año el tamaño de los circuitos integrados y chips se ha ido reduciendo para aumentar la potencia y eficiencia de los ordenadores, llegando a un punto de escala de nanómetros. Llegados a esta situación no se puede seguir disminuyendo el tamaño de los componentes, ya que llegaría

un momento en el que las interacciones cuánticas de los átomos pondrían en peligro la fiabilidad de nuestros ordenadores. Además, tampoco se puede subir la frecuencia del reloj de los ordenadores porque el incremento de la temperatura los haría inestables y se podrían dañar. Esto quiere decir que los procesadores convencionales han llegado a su límite. De hecho, la velocidad máxima de un único procesador sin overclock se ha mantenido estable alrededor de los 4Ghz desde 2004. El Intel Pentium 4 HT a 4,00 GHz salió a inicios de 2004 y uno de los mejores procesadores actuales es el Intel Core i7-8700K a 3,70 Ghz con Turbo Boost a 4,7 Ghz. Esta situación ha forzado el desarrollo de un nuevo tipo de ordenadores y computación capaces de seguir mejorando las prestaciones.

La idea de computación cuántica surge en 1981 cuando Paul Benioff [Ben80] expuso su teoría para aprovechar las leyes cuánticas en el entorno de la computación. Cuando intentaron simular el comportamiento de estados físicos de partículas entrelazadas se dieron cuenta de que necesitaban una cantidad de computación exponencial en relación con el número de partículas. Los ordenadores cuánticos, a diferencia de los ordenadores convencionales, trabajan con qubits en vez de con bits. Los bits pueden tomar el valor 0 o 1, en cambio los qubits pueden tener distintos estados y pueden estar en superposición, lo que conlleva mayor información. El número de qubits indica la cantidad de bits que pueden estar en superposición y, además, la cantidad de información es exponencial con respecto al número de qubits. Por otra parte, otra ventaja que tiene la computación cuántica es que hay operaciones lógicas que solo se pueden hacer con un qubit y no con un bit, luego si trabajamos con qubits se tiene la opción de trabajar con nuevos tipos de algoritmos. Hay que tener en cuenta que los algoritmos cuánticos no son como los tradicionales, en el sentido de que no siempre se les puede dar el input deseado, ya que no se puede crear qubits con un estado determinado.

Representar físicamente un qubit no es nada fácil, de hecho no hay una manera única o estándar de representarlo. Hoy en día, hay una gran investigación para descubrir sistemas físicos capaces de representar uno o varios qubits. Dichos sistemas se englobarían dentro de la física cuántica, ya que la noción de qubit tiene asociada la noción de superposición, entrelazamiento y aleatoriedad, que justamente se dan en esta rama de la física. Matemáticamente un qubit puede representarse por un vector de módulo 1 en el espacio de Hilbert complejo bidimensional. Los estados básicos son $|0\rangle$ y $|1\rangle$, aunque también puede tomar un estado combinado $\alpha|0\rangle + \beta|1\rangle$. En dichos espacios de Hilbert se pueden definir operadores con ciertas propiedades, los cuales dan lugar a puertas lógicas que se pueden utilizar en distintos tipos de algoritmos, por lo que hay un gran estudio en este tipo de espacios y es el Análisis Funcional quien se encarga de ello.

Ahora veamos cuál es el estado actual de los ordenadores cuánticos. Durante la década del 2000 se empiezan a desarrollar ordenadores con varios qubits, los cuales pueden hacer operaciones básicas. Por ejemplo, se tiene el famoso caso de la factorización del número 15 con el ordenador desarrollado por IBM de 7 qubits. Hoy en día uno de los ordenadores cuánticos más potentes es el D-Wave Two, desarrollado por D-Wave Systems, con "512 qubits". Se supone que este ordenador es 4000 veces superior a un microprocesador Intel Xeon a 2,9 GHz. Decimos se supone, porque hay una gran controversia sobre esto, ya que este ordenador solo es capaz de entrelazar 8 qubits, lo que quiere decir que es un ordenador cuántico de 64 procesadores y cada procesador es capaz de trabajar con 8 qubits. Además, los test realizados para comparar dichos procesadores dependen mucho de qué procesos se lleven a cabo para la comparación, ya que se dice que el algoritmo utilizado en el PC convencional a comparar estaba muy mal optimizado. Por tanto, no está claro cuál es la potencia real, lo que sí se puede afirmar es que, de momento, esto no pone en peligro los sistemas criptográficos actuales.

En 2012 IBM anunció que tenía un procesador cuántico estable, que podría empezar a comercializar en serie en una década. Si fuese así, un gran número de investigadores tendrían acceso a este tipo de ordenadores, a diferencia de ahora que solo un número muy limitado de personas tienen acceso a estos. Uno de los problemas que tienen son los errores producidos por las interacciones cuánticas, aunque en determinados sistemas se pueden eliminar dichos errores. En junio de 2014 un grupo de investigadores austriacos y españoles de la UCM han conseguido entrelazar 7 qubits sin errores cuánticos [NMM+14], lo que supone un gran avance.

Evidentemente, el precio y tamaño de estos ordenadores es desorbitado, por ejemplo, el D-Wave Two es más grande que una habitación y es propiedad de la NASA y Google, que lo compraron por 15 millones de dólares, lo que quiere decir que deben de pasar aún bastantes años hasta que la computación cuántica sea fiable, potente y económica. En el congreso mundial sobre criptografía post-cuántica del año 2016 celebrado en Japón, el instituto Nacional de Estándares y Tecnología Americano (NIST) [NIS] estimo que dentro de 20 años se podrán tener ordenadores cuánticos lo suficientemente potentes como para romper el RSA-2048. Por otra parte, será muy difícil que el día de mañana estos ordenadores estén en todas las casas, por la complejidad y tamaño de los componentes. Además, hay aparatos como los móviles, routers o tarjetas electrónicas que será muy difícil que incorporen procesos cuánticos, por lo que es necesario desarrollar sistemas criptográficos clásicos que se puedan implementar en cualquier dispositivo y que también sean potencialmente resistentes a ataques con ordenadores cuánticos. De esto se encarga la criptografía

post-cuántica.

Paralelamente al desarrollo de los sistemas de criptografía post-cuántica, se está trabajando y estudiando ataques a estos mismos. Estos ataques pueden ser tanto generales, para un grupo amplio de criptosistemas, como específicos, enfocados a las vulnerabilidades particulares de un sistema concreto. Actualmente hay una gran investigación sobre un tipo de ataque general, los llamados ataques algebraicos, como el Algoritmo F4 [Fau99]. Esto surge por dos motivos, el primero, porque son muy utilizados en el estudio de sistemas multivariables cuadráticos y el segundo porque sistemas de clave privada, como el AES-128 [DR02], se pueden reformular como sistemas de ecuaciones no lineales [Bar09]; que podrían ser atacados por este tipo de algoritmos, por tanto la seguridad de dichos sistemas está relacionada con la seguridad del sistema de ecuaciones asociado. Por lo cuál, en el desarrollo de sistemas post-cuánticos se tiene que tener en mente este tipo de ataques para que nuestro sistema no presente vulnerabilidades.

1.2. Introducción a la Criptografía Multivariable

En esta sección se presentan las ideas generales de los criptosistemas MPKC (Multivariate Public Key Cryptography [DGS06]) y cómo se construyen, para presentar el marco donde se trabajará, y en el capítulo 4 retomaremos estos sistemas más en detalle. Los sistemas MPKC son uno de los tipos de sistemas post-cuánticos más estudiados actualmente. Como en todos los sistemas de clave pública, la dificultad de romper un sistema MPKC radica en la dificultad de resolver un problema. Dicho problema consiste en la resolución de sistemas de ecuaciones no lineales en varias variables, el cual se cree que es resistente al Algoritmo de Shor. Veremos la definición de este problema y presentaremos un sistema de ejemplo. Primeramente definamos varios objetos para fijar la notación.

Sea $\mathbb{K}$ un cuerpo finito. Sea ψ el siguiente homomorfismo:

$$\begin{aligned} \psi_{\mathbb{K}} : \mathbb{Z} &\longrightarrow \mathbb{K} \\ 1 &\longrightarrow 1_{\mathbb{K}} \end{aligned}$$

Como $\mathbb{Z}$ es un DIP y $\mathbb{K}$ es finito se tiene que el $\ker \psi_{\mathbb{K}} = (p)$, con $p \neq 0$, ya que sino, $\mathbb{K}$ no sería finito.

Definición 1.1. Sea $\mathbb{K}$ un cuerpo finito. Definimos la característica de $\mathbb{K}$ como $char(\mathbb{K}) = p > 0$, donde p es el generador del $\ker \psi_{\mathbb{K}}$.

Observación 1.2. Como $\mathbb{K}$ es un dominio de integridad y $\mathbb{Z}/ker\,\psi \hookrightarrow \mathbb{K}$, necesariamente $char(\mathbb{K}) = p$ es un número primo y $\mathbb{Z}_p \simeq \mathbb{F}_p \hookrightarrow \mathbb{K}$.

En esta situación se tiene que, $\mathbb{K}$ es un $\mathbb{F}_p$-espacio vectorial de dimensión n y, por tanto, $|\mathbb{K}| = p^n$. Se puede ver que $\mathbb{K}^*$ es un grupo cíclico de $p^n - 1$ elementos, por tanto, todo elemento de $\mathbb{K}$ verifica la ecuación

$$x^{p^n} - x = 0.$$

Como ya se ha visto, se tiene que si $\mathbb{K}$ es una extensión de grado n, entonces $\mathbb{K}$ tiene estructura de $\mathbb{F}_p$-espacio vectorial. Además, se verifica que la aplicación $x \to x^p$ es un automorfismo de $\mathbb{K}$.

Hay determinadas bases que pueden facilitar los cálculos de las operaciones en el cuerpo. Supongamos que:

$$\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(f),$$

con f un polinomio irreducible de grado n sobre $\mathbb{F}_p[x]$. Entonces si α es una raíz de f, se puede tomar como base de $\mathbb{F}_{p^n}$ sobre $\mathbb{F}_p$ el conjunto $\left\{1, \alpha, \ldots, \alpha^{n-1}\right\}$. Con este tipo de bases los cálculos de las multiplicaciones, entre elementos del cuerpo, se hace de manera muy rápida si se sabe sus coordenadas respecto de dicha base. Ya que se tiene $\alpha^r \cdot \alpha^s = \alpha^{r+s}$, si $r + s < n$, y en caso contrario se tiene $\alpha^r \cdot \alpha^s = \alpha^n \cdot \alpha^{r'+s'}$, con $r' + s' < n$; y solo debemos ayudarnos de la expresión $f(\alpha) = 0$ para obtener el valor de α^n. De hecho se tiene el siguiente resultado.

Proposición 1.3. *Sea $\mathbb{K}$ un cuerpo finito de característica p con $q = p^n$ elementos. Dos elementos de $\mathbb{K}$ pueden multiplicarse en $O(\ln^2 q)$ operaciones.*

Demostración. [Sho09] □

Sobre los cuerpos finitos se puede factorizar rápidamente polinomios de una sola variable. Para factorizar un polinomio, sobre un cuerpo $\mathbb{K}$ de q elementos, se utiliza el Algoritmo de Berlekamp [Sho09], el cual tiene complejidad de

$$O(l^3 + l^2 \ln l \ln q)$$

si el polinomio f es mónico, libre de cuadrados y de grado l.

1.2.1. El problema $\mathcal{MQ}$

Cuando en criptografía se diseña un criptosistema siempre se piensa en un problema difícil para basar el sistema en él. Por ejemplo, el RSA se basa en la dificultad de factorizar un número natural compuesto. Lo que interesa es un sistema con una función unidireccional basada en un problema muy difícil. El termino muy difícil hace referencia a que hasta la fecha no se sepa que dicho problema pertenezca a la clase P de problemas polinomiales. Dicha función de un sentido ha de contener una trampa para luego poder invertirla. En el RSA la función de un sentido es la exponenciación en un grupo de tamaño n, y la trampa es que sabiendo la descomposición de n se puede obtener la inversa de dicha función. Como es evidente, al introducir la trampa la dificultad del sistema no es exactamente la misma que la dificultad del problema en el que está basado nuestro sistema. Ahora presentaré el Problema $\mathcal{MQ}$, un problema en el que se basan muchos sistemas de Criptografía Post-Cuántica.

Resolver sistemas de m ecuaciones cuadráticas en n variables es uno de los retos de los criptoanalistas algebraicos. Este problema es muy importante en criptografía ya que muchos sistemas basan su eficacia en él, incluso sistemas muy utilizados hoy en día. Veamos cuál es su definición formal.

Definición 1.4 (Problema $\mathcal{MQ}$)**.** Sean $m, n \in \mathbb{N}$ y $\mathbb{F}_q$ un cuerpo finito de q elementos. Denotamos por

$$P : \mathbb{F}_q^n \to \mathbb{F}_q^m$$

a una aplicación polinomial cuadrática de varias variables. Definimos

$$PQ := \left\{(P, y) : y \in \mathbb{F}_q^m \text{ y } \exists x \in \mathbb{F}_q^n \text{ tal que } P(x) = y\right\}.$$

El problema $\mathcal{MQ}$ correspondiente es, dado un par (P, y) decidir si, $(P, y) \in PQ$, o lo que es lo mismo, decidir si $y \in Im(P)$. En caso afirmativo calcular un x tal que $P(x) = y$, es decir, encontrar una solución del sistema.

En general el problema $\mathcal{MQ}$ sobre cuerpos finitos es NP-completo [Bar09, Kob98]. Aunque, evidentemente, la complejidad de un problema concreto está relacionada con los valores m, n y la estructura de los polinomios involucrados. Se sabe que el problema $\mathcal{MQ}$ se puede resolver en tiempo polinomial sobre un cuerpo finito $\mathbb{F}_q$ cuando el sistema es muy sobredefinido, es decir, cuando $m \geq n(n+1)/2 + n$.

Además, de las ecuaciones del propio problema, se pueden añadir las ecuaciones del cuerpo

$$x_i^q - x_i = 0 \quad i = 1, \ldots, n$$

para optimizar la búsqueda de una solución. Esto solo servirá cuando el tamaño del cuerpo sea pequeño, ya que así se podrán limitar los exponentes de los monomios durante la búsqueda de una Base de Groebner [CLO07]. Si el tamaño del cuerpo es grande la limitación de los exponentes no afectará y, por tanto, estaremos añadiendo ecuaciones que estorban en los cálculos intermedios, esto se puede comprobar fácilmente de manera experimental. En determinadas ocasiones un atacante se puede ayudar de la estructura del sistema para resolverlo. Por ejemplo, cuando los sistemas son sobredefinidos hay algoritmos específicos, como el XL [CKPS00], que pueden ayudar a resolver parcial o totalmente el sistema. También hay algoritmos como el F4 o la versión de Gebauer y Möller del Algoritmo de Buchberger [MMT92], que en determinados casos son eficientes, pero que en general no sirven para resolver sistemas aleatorios con n suficientemente grande.

Gracias a la complejidad del problema $\mathcal{MQ}$ existen varios criptosistemas basados en él. La idea es construir una función de cifrado de un sentido que tenga una trampa, para poder leer el texto cifrado, y que dicha función de cifrado sea una aplicación polinomial cuadrática. Evidentemente, si pudiésemos resolver el problema $\mathcal{MQ}$ en tiempo polinomial el sistema estaría roto. Pero al revés no sería cierto, ya que al introducir "la trampa" en la función de codificación, nos estaremos quedando con un tipo especial de problemas $\mathcal{MQ}$. Justamente ésta es la razón por la que muchos de los sistemas basados en este problema se han roto, ya que se han utilizado las características particulares de los sistemas para resolverlos, como por ejemplo el cifrado de la mochila [MH78]. Una de las buenas características que tiene el RSA, a diferencia de estos sistemas, es que si pudiésemos tener un algoritmo polinomial que rompiese RSA, entonces se tendría un algoritmo polinomial probabilístico que factorizaría n. Esto muestra que romper RSA es casi tan difícil como factorizar un número natural.

En general, se toman polinomios cuadráticos y no de mayor grado por una cuestión de eficiencia en el proceso de encriptado y para reducir el tamaño de la clave [DGS06, BBD09, Bar09]. De hecho, incluso tomando polinomios cuadráticos, el tamaño de la clave es mucho mayor que el de otros sistemas de criptografía asimétrica.

En 1979 se probó que el problema de decisión MQ es NP-Completo. La manera de hacerlo es ver que cualquier ecuación del 3-SAT (problema NP-Completo de sentencias lógicas) puede ser reducida a un sistema de ecuaciones cúbicas [Bar09]. Por otra parte cualquier sistema de ecuaciones polinomiales se puede poner como un sistema de ecuaciones cua-

dráticas simplemente añadiendo ecuaciones y variables. Por lo tanto, cualquier problema 3-SAT puede ponerse como un problema $\mathcal{MQ}$. Por otra parte, si hubiese un algoritmo capaz de resolver el problema de decisión MQ entonces se podrían resolver los sistemas de ecuaciones polinomiales sin más que fijar una de las variables y preguntar si el sistema $P(x_1, ..., c, ..., x_n)$ tiene solución. Esto asegura que resolver sistemas polinomiales es NP, pero por otra parte en determinados casos existen algoritmos que resuelven en tiempo polinomial ciertos sistemas.

1.2.2. Sistemas MPKC

Los sistemas MPKC (Multivariate Public Key Cryptography) [BBD09, DGS06] son un tipo de sistemas de clave pública basados en el problema $\mathcal{MQ}$. Se cree que dichos sistemas serían resistentes a ataques con ordenadores cuánticos. El primer sistema MPKC presentado fue el Imai-Matsumoto en el año 1988, el cual se puede ver en [MI88]. Como ya se ha comentado, al introducir la trampa en la función de encriptado no está garantizado que la complejidad para resolver dichos sistemas sea NP. La forma estándar de construcción de estos sistemas es la siguiente:

1. Se fija un cuerpo $\mathbb{F}_q$.
2. Se fija una aplicación cuadrática $\mathcal{Q}$ (o aplicación polinomial) inyectiva.
3. Se escogen dos aplicaciones afines invertibles T y S.
4. Se construye la clave pública $P = T \circ \mathcal{Q} \circ S$.

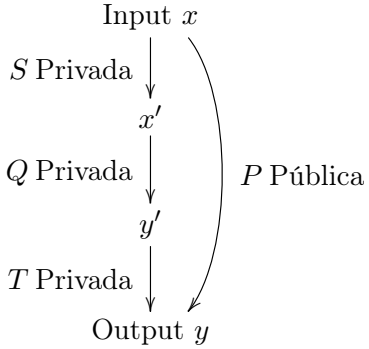

Figura 1.2.1: Esquema general de los MPKC

El resultado final es un sistema no lineal de ecuaciones en varias variables, de forma que el sistema de ecuaciones será la clave pública P. Para generar el texto cifrado y solo hay que evaluar dicho sistema, $y = P(x)$, lo cual es muy eficiente. La aplicación $\mathcal{Q}$ se puede invertir fácilmente o se puede computar fácilmente la preimagen de un valor, por tanto conociendo la clave privada T, S y $\mathcal{Q}$ se puede conocer el texto claro asociado a un texto cifrado. Además, como pasa en el proceso de encriptado, el proceso de desencriptado es muy eficiente. Las aplicaciones S y T se introducen únicamente para ocultar la estructura de la aplicación Q, así aunque la aplicación Q sea muy sencilla o simple el sistema de ecuaciones P parecerá un sistema de ecuaciones aleatorio.

En este tipo de sistemas la elección del cuerpo es importante. No interesa que el tamaño del cuerpo sea muy grande para que el tamaño de la clave no se dispare, pero por otra parte, interesa que el tamaño del cuerpo no sea muy pequeño para que el atacante no se pueda ayudar de las ecuaciones

$$x_i^q - x_i = 0 \quad i = 1, \ldots, n$$

para resolver el sistema.

Hay dos maneras de atacar estos sistemas:

- Ataques generales: son los ataques que resuelven el sistema de ecuaciones cuadráticas resultante, como por ejemplo mediante Bases de Groebner o búsqueda exhaustiva de soluciones.
- Ataques de estructura: son los ataques que se basan en la propia estructura del sistema para resolverlo, como por ejemplo el ataque MinRank[GC00].

Uno de los temas que recibe mucha atención es la complejidad de los algoritmos que se encargan de resolver sistemas de ecuaciones no lineales. Hay algoritmos que funcionan bien en determinadas situaciones, pero no se sabe cuál es su complejidad exacta. Alguno de estos algoritmos, como el F4 [Fau99], en general no son capaces de resolver este tipo de sistemas con n suficientemente grande. No porque el tiempo de ejecución se dispare, sino porque la memoria empleada en dichos algoritmos aumenta exponencialmente con el tiempo de ejecución, por lo que en determinadas ocasiones el algoritmo se para al exceder la memoria del ordenador.

Una de las principales características de los sistemas MPKC es que son rápidos en el proceso de encriptado, pero en cambio el tamaño de su clave pública es elevada. Véase un ejemplo en la figura 1.2, en donde los 4 últimos sistemas son MPKC.

Sistema	ClavSec	ClavPubl	GenrClav	Descript	Encript
RSA-1024	128 B	320 B	2.7 seg	84 ms	2 ms
ECDSA-$\mathbb{F}_{2^{163}}$	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
PMI+$(136, 6, 18, 8)$	5.5 kB	165 kB	1.1 seg	1.23 ms	0.18 ms
Rainbow$(28, 18, 12, 12)$	24.8 kB	22.5 kB	0.3 seg	0.43 ms	0.4 ms
Rainbow$(24, 24, 20, 20)$	91.5 kB	83 kB	1.6 seg	0.93 ms	0.74 ms
QUARTZ	71 kB	3.9 kB	3.1 seg	11 seg	0.24 ms

Cuadro 1.2: Comparación de sistemas MPKC en un Pentium III 500[BBD09]

Normalmente estos sistemas se construyen sobre los cuerpos $\mathbb{F}_{2^p}$, con $p \approx 8$. Esto es muy útil, ya que para multiplicar dos números se escoge un generador g de $\mathbb{F}_{2^p}^*$, y posteriormente con la tabla de logaritmos se calcula

$$x \cdot y = g^{(log_g x + log_g y)}.$$

Lo que hace que la multiplicación sea muy rápida sobre estos cuerpos, puesto que realmente se está haciendo una suma.

Uno de los retos de los investigadores que trabajan con sistemas MPKC es convencer a la industria de que dichos sistemas son seguros. En estos últimos años han aparecido muchos de estos sistemas, pero en poco tiempo han sido rotos por algoritmos muy ingeniosos, por lo que se puede ser susceptible con este tipo de sistemas. Ahora veamos uno de los primeros sistemas MPKC, muy popular por su elegancia, el cual muestra la esencia y las generalidades de este tipo de sistemas.

1.2.3. Little Dragon

Este criptosistema fue introducido por Jacques Patarin en 1996 [Pat96b], después de haber roto el sistema Imai-Matsumoto. Se presenta este sistema para mostrar al lector la idea de construcción de los MPKC, es decir, la idea de buscar una aplicación polinomial cuadrática con una puerta trasera, que se pueda utilizar para calcular la preimagen de un punto.

Sea $\mathbb{K}$ una extensión de grado n del cuerpo finito $\mathbb{F}_q$, y sea $\mathcal{B} = \{\beta_1, \ldots, \beta_n\}$ una base de $\mathbb{K}$ como $\mathbb{F}_q$-espacio vectorial. Los textos claros y los textos cifrados se representarán por n-tuplas, $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ respectivamente. Además, durante el proceso de encriptado se trabaja con dos vectores intermedios $x' = (x'_1, \ldots, x'_n), y' = (y'_1, \ldots, y'_n) \in \mathbb{F}_q^n$ de manera que $x' = Ax$ y $y' = By$, donde A y B son dos matrices $n \times n$ invertibles con entradas en $\mathbb{F}_q$. El método de encriptado es muy sencillo, veamos cual es el

procedimiento. Lo primero que se hace es elegir al azar un $h \in \mathbb{N}$ tal que $0 < h < q^n$ y que verifique:

$$\gcd(h, q^n - 1) = 1,$$

y que existan $\theta_1, \theta_2 \in \{1, \ldots, n-1\}$ tales que:

$$h = q^{\theta_1} + q^{\theta_2} - 1.$$

Aunque a priori el elemento h sea secreto, al haber solamente $O(n^2)$ posibilidades de elegir h, se puede asumir que un atacante puede probar con todos los posibles h y, por tanto, se puede pensar que puede conocer dicho valor. Una vez elegido h, el usuario selecciona al azar las matrices A y B. Hay dos formas de encriptar un texto claro. La primera involucra directamente la exponenciación, y consiste en los siguientes pasos

1. Se calcula $x' = Ax$.
2. Sea $u = x'_1\beta_1 + \cdots + x'_n\beta_n \in \mathbb{K}$, se calcula $v = u^h$.
3. Sea y' las coordenadas de v respecto de la base $\mathcal{B}$, el texto encriptado y vendrá dado por $y = B^{-1}y'$.

Para utilizar este proceso de encriptado se necesita saber h, A, B y $\mathcal{B}$, pero hay otra manera de calcular el mismo texto cifrado sin necesidad de saber ninguno de estos parámetros. Como $v = u^h$, por definición de h se tiene que:

$$uv = u^{q^{\theta_1}} u^{q^{\theta_2}}. \tag{1.2.1}$$

Como ya se sabe, las potencias q^{θ_k}-ésimas en $\mathbb{K}$ son aplicaciones lineales sobre $\mathbb{F}_q$. Sean pues $P^1 = (p^1_{ij})$ y $P^2 = (p^2_{ij})$ las matrices asociadas a dichas aplicaciones lineales, y $(m^{ij}_1, \ldots, m^{ij}_n)$ las coordenadas del elemento $\beta_i\beta_j$ respecto de la base $\mathcal{B}$. Entonces la ecuación 1.2.1 se puede escribir como

$$\begin{aligned} \sum_{1 \le i,j \le n} x'_i y'_j \beta_i \beta_j &= \left(\sum_{i=1}^{n} x'_i \beta_i^{q^{\theta_1}} \right) \left(\sum_{j=1}^{n} x'_j \beta_j^{q^{\theta_2}} \right) \\ &= \left(\sum_{1 \le i,k \le n} p^1_{ik} x'_i \beta_k \right) \left(\sum_{1 \le j,l \le n} p^2_{jl} x'_j \beta_l \right), \end{aligned}$$

o lo que es lo mismo

$$\sum_{1\leq i,j\leq n} m_r^{ij} x'_i y'_j = \sum_{1\leq i,j,k,l\leq n} m_r^{kl} p_{ik}^1 p_{jl}^2 x'_i x'_j \quad r = 1,\ldots,n.$$

Como se tiene que $x' = Ax$ y $y' = By$, sustituyendo x'_i por $\sum a_{it}x_t$, e y'_i por $\sum b_{it}y_t$, se obtiene un sistema de n ecuaciones cuadrático. Dicho sistema tendrá la siguiente forma:

$$\sum_{1\leq i,j\leq n} c_{ij}^k x_i y_j + \sum_{1\leq i\leq j\leq n} d_{ij}^k x_i x_j = 0 \quad k = 1,\ldots,n, \tag{1.2.2}$$

con $c_{ij}^k, d_{ij}^k \in \mathbb{K}$. Este conjunto de ecuaciones forman la clave pública y la clave privada será el conjunto $\{h, A, B, \mathcal{B}\}$. Por tanto, para obtener el texto cifrado solo se tiene que introducir el valor de las x_i's en la ecuación 1.2.2, para luego resolver el sistema lineal resultante, obteniendo así el valor de las y_i's. De esta manera, un atacante que conociese un texto cifrado se encontraría con un sistema de ecuaciones cuadrático. La manera de desencriptado, conociendo la clave privada, es muy sencilla, solo hay que seguir los siguientes pasos:

1. Se calcula $y' = By$.
2. Sea $v = y'_1\beta_1 + \cdots + y'_n\beta_n \in \mathbb{K}$, se calcula $u = v^{h'}$, donde $h'h \equiv 1 \text{ mód } q^n - 1$.
3. Sea x' las coordenadas de u respecto de la base $\mathcal{B}$, el texto claro x vendrá dado por $x = A^{-1}x'$.

Gracias a cómo se ha construido el sistema, se puede desencriptar un mensaje rápidamente sabiendo la clave privada. Pero si no se conoce, nos enfrentaremos a un sistema de ecuaciones cuadrático, el cual a priori no presenta ninguna estructura. Como poco después se vio[Pat96b], este criptosistema no es seguro. Veamos cuales son las ideas principales para romper dicho sistema.

Sea $v \in \mathbb{K}$ y $\overline{v} \in \mathbb{F}_q^n$ las coordenadas de v respecto de la base $\mathcal{B}$. La idea del criptoanálisis es buscar una función bilineal $*$ que se comporte como el producto en $\mathbb{K}$, es decir, que exista un $\lambda \in \mathbb{K}$ de manera que $*$ haga el siguiente diagrama conmutativo

$$\begin{array}{ccc} (\overline{v}_1, \overline{v}_2) & \to & \overline{v}_1 * \overline{v}_2 \\ \downarrow & \circlearrowleft & \downarrow \\ (v_1, v_2) & \to & \lambda \cdot v_1 \cdot v_2 \end{array}$$

Brevemente decir que dicha aplicación se encuentra gracias al hecho de que para todo $\lambda \in \mathbb{K}$ se verifica que $\lambda(v_1 v_2) = v_1(\lambda v_2)$. Una vez se tenga dicha aplicación bilineal el sistema estará roto, pues se verifica que:

$$x = Cy^{(h')}$$

donde

$$y^{(h')} = \overbrace{y * \cdots * y}^{h'}$$

y C es una matriz $n \times n$ con entradas en $\mathbb{F}_q$, que se puede calcular a partir de textos claros y textos cifrados generados por nosotros mismos.

Capítulo 2

Ataques Algebraicos

En este capítulo se presentarán los ataques algebraicos más relevantes que se pueden utilizar en el marco de los sistemas MPKC. Primero se verá la relinealización y el algoritmo XL [CKPS00], luego se verá en detalle el algoritmo F4 y se mostraran resultados interesantes sobre su complejidad.

Cualquier texto cifrado de un sistema MPKC puede ser descifrado resolviendo el sistema de ecuaciones resultante. Un primer enfoque sería probar uno de los siguientes métodos:

1. Si el cuerpo sobre el que se trabaja es pequeño, en relación con el número de variables, podemos ir probando uno a uno los posibles valores hasta encontrar una solución del sistema.

2. En cambio si el cuerpo sobre el que se trabaja es grande, en relación con el número de variables, lo mejor es utilizar el Algoritmo de Buchberger, o cualquier otro algoritmo de cálculo de Bases de Groebner. La manera de proceder sería la siguiente: se escoge un orden monomial, como el lexicográfico, de manera que durante la ejecución del Algoritmo de Buchberger se llega a una ecuación que solo depende de una variable. Luego, por el Algoritmo de Berlekamp, se resuelve dicha ecuación en el cuerpo y se encuentra el valor de dicha variable. Ahora se volvería a aplicar el mismo procedimiento con las $n-1$ variables restantes, por lo que en n pasos se llega a una solución del sistema.

3. Utilizar una mezcla de los dos métodos anteriores. Para ello se fijaría de manera aleatoria el valor de ciertas variables y se procedería a la resolución del sistema resultante. Si no se puede obtener una solución del sistema inicial se volvería a repetir el proceso fijando el valor de otras variables. Así hasta encontrar una solución.

La pega del primer método es que su complejidad es exponencial en el número de variables, por lo que en la práctica no sería eficiente. La pega del segundo método es que el Algoritmo de Buchberger empieza a fallar cuando $n \geq 15$, ya que la cantidad de memoria necesaria es enorme, además su complejidad es doblemente exponencial. Como luego se verá, hay algoritmos más eficientes para el cálculo de las Bases de Groebner, como el F4, cuya complejidad es $O(2^{3n})$ en la mayoría de los casos. Veamos ahora otros métodos básicos, pero efectivos en determinadas circunstancias, capaces de resolver sistemas de ecuaciones polinomiales.

2.1. Linealización y Relinealización

Idea de la Linealización

En esta sección se verán dos algoritmos concretos para la resolución de sistemas sobredefinidos y cuadráticos. En general no tendrían porqué ser cuadráticos, pero nos centraremos en este caso para simplificar los razonamientos. El primero de estos algoritmos es la Linealización. Como su nombre indica lo que se hará será linealizar el sistema. La idea de este algoritmo es cambiar cada monomio $x_i x_j$ por una nueva variable y_{ij}, y cada monomio x_i por la variable y_i. Con esto se pasa a un sistema lineal que se pueda resolver por Eliminación Gaussiana. Partimos de un sistema que tiene la siguiente forma:

$$\sum_{1 \leq i,j, \leq n} a_{i,j}^k x_i x_j + \sum_{1 \leq i \leq n} b_i^k x_i + c^k = 0 \quad k = 1, \ldots, m,$$

con $a_{i,j}^k, b^k \in \mathbb{F}_q$. Si se verifica que el sistema tuviese $m = n(n+1)/2 + n$ ecuaciones independientes, aplicando la Linealización se llega a un sistema lineal de rango máximo con n' variables y $n(n+1)/2 + n$ ecuaciones donde

$$n' \leq \frac{n(n+1)}{2} + n = \binom{n+1}{2} + n = |\{x_i x_j : 1 \leq i, j \leq n\}| + |\{x_i : 1 \leq i \leq n\}|.$$

En esta situación se puede resolver rápidamente el sistema lineal y, como consecuencia, resolver el sistema inicial. En general solo se necesita que el número de ecuaciones linealmente independientes sea igual al número de monomios distintos del sistema.

Evidentemente, en determinadas situaciones se puede resolver el sistema inicial aunque el número de ecuaciones no sea tan elevado. Como se ha visto, este algoritmo funciona bien en el caso de que $m \approx n^2/2$, ya que en dicha situación se puede resolver parcial o totalmente

solo con el Método de Gauss. Aunque no se haya comentado, la Eliminación Gaussiana es un algoritmo polinomial de complejidad $O(n^{2,37})$. En general, el número de ecuaciones no es tan grande en relación con el número de incógnitas, por tanto este método no se puede aplicar a la mayoría de sistemas MPKC. El siguiente algoritmo es una mejora de la Linealización, aunque fundamentalmente son muy parecidos.

Idea de la Relinealización

Veamos ahora como funciona la Relinealización [CKPS00]. Supongamos que una vez aplicada la Linealización a nuestro sistema no se puede resolver, ya que se desconoce el valor de ciertas variables $y_{\alpha_1}, \ldots, y_{\alpha_k}$. En esta situación nos gustaría añadir nuevas ecuaciones, que se verifiquen trivialmente y que sean independientes con las ecuaciones que ya teníamos, para volver a aplicar Linealización y resolver el sistema. La idea de la Relinealización es añadir las ecuaciones que se verifican trivialmente por la conmutatividad de la multiplicación, dichas ecuaciones son

$$(x_{\sigma(1)}x_{\sigma(2)})\ldots(x_{\sigma(2k-1)}x_{\sigma(2k)}) = (x_{\tau(1)}x_{\tau(2)})\ldots(x_{\tau(2k-1)}x_{\tau(2k)}),$$

que corresponden con las ecuaciones, en el sistema ya linealizado, con

$$y_{\sigma(1)\sigma(2)}\cdots y_{\sigma(2k-1)\sigma(2k)} = y_{\tau(1)\tau(2)}\cdots y_{\tau(2k-1)\tau(2k)},$$

donde $\sigma, \tau \in S_{2k}$. Al número $2k$ se le llama el grado de Relinealización. Nos centraremos en la Relinealización de grado 4, ya que muestra la esencia y las características de este método sin necesidad de muchos detalles. Una vez se tengan dichas ecuaciones, se considera el sistema resultante de quedarnos solo con las ecuaciones linealmente independientes

$$y_{\sigma(1)\sigma(2)}y_{\sigma(3)\sigma(4)} - y_{\tau(1)\tau(2)}y_{\tau(3)\tau(4)} = 0 \quad \sigma, \tau \in S_4'.$$

Una vez hecho esto, sustituimos el valor de las incógnitas halladas anteriormente. Se obtiene así un sistema de ecuaciones no lineales y cuadrático, que solo depende de las variables $y_{\alpha_1}, \ldots, y_{\alpha_k}$. Finalmente se debe aplicar Linealización a este nuevo sistema para resolverlo. En la práctica cuanto menor sea el número $m - n$ mayor debería ser el grado de la Relinealización que se fuese a utilizar. Veamos un análisis de la Relinealización de grado 4.

Análisis de la Relinealización

Partimos de un sistema cuyas ecuaciones son de la forma

$$\sum_{1\leq i,j,\leq n} a_{i,j}^k x_i x_j + \sum_{1\leq i\leq n} b_i^k x_i + c^k = 0 \quad k = 1,\ldots,m.$$

Supongamos que se tienen n variables y $m = \epsilon n^2$ ecuaciones, con $\epsilon \in \mathbb{R}_{>0}$. Lo primero que se tiene que hacer es linealizar el sistema para obtener un sistema lineal de

$$n' = \binom{n+1}{2} + n = \frac{n(n+1)}{2} + n$$

variables y m ecuaciones. Por Gauss se resuelve el sistema y, sin pérdida de generalidad, se tiene que $y_1,\ldots,y_m$ dependerían de los parámetros $y_{m+1},\ldots,y_{n'}$. Por tanto, solo se tiene que encontrar el valor de las últimas $n'-m$ variables ayudándonos de un nuevo sistema, cuyas ecuaciones son las descritas anteriormente. En el caso de grado 4 se tienen que introducir las ecuaciones de la forma

$$y_{ab}y_{cd} = y_{a'b'}y_{c'd'},$$

quedándonos solo con las ecuaciones que sean linealmente independientes. Veamos cuantas ecuaciones de esta forma son linealmente independientes. Se dirá que $(ab)(cd) = (a'b')(c'd')$ si, y solo si, $y_{ab}y_{cd} = y_{a'b'}y_{c'd'}$. Con 4 índices distintos a,b,c,d se pueden generar dos ecuaciones linealmente independientes, ya que

$$(ab)(cd) = (ac)(bd) = (ad)(bc),$$

y cualquier otra ordenación de los índices daría una ecuación linealmente dependiente, luego se pueden formar $2\binom{n}{4}$ ecuaciones independientes. Supongamos que solo se tiene 3 índices distintos, entonces hay 3 posibilidades, a,a,b,c o a,b,b,c o a,b,c,c. El primer caso genera una ecuación linealmente independiente, ya que

$$(aa)(bc) = (ab)(ac),$$

y cualquier otra ordenación genera ecuaciones dependientes. Procediendo igual en los demás casos, se tiene que con 3 índices distintos pueden formar $3\binom{n}{3}$ ecuaciones linealmente independientes. Si ahora se supone que se tienen solo 2 índices distintos, se tendrían 2 posibilidades, o bien a,a,a,b o bien a,a,b,b. El primer caso no genera ninguna ecuación y

el segundo genera solamente una, ya que

$$(aa)(bb) = (ab)(ab).$$

Por lo cual, con 2 índices distintos se pueden formar $\binom{n}{2}$ ecuaciones linealmente independientes. Al final se obtiene que se pueden añadir

$$2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} = \frac{1}{12}(n^4 - n^2)$$

ecuaciones linealmente independientes. Entonces, a partir de este proceso, se ha formado un sistema de

$$m' = \frac{1}{12}(n^4 - n^2)$$

ecuaciones con n' variables. Como las m primeras incógnitas dependen linealmente de las demás, sustituyendo su valor se llega a un sistema de m' ecuaciones y $n'' = n' - m$ variables. Si se vuelve a aplicar Linealización, se obtiene un sistema lineal de m' ecuaciones y $\binom{n''+1}{2}+n''$ variables, con todas sus ecuaciones linealmente independientes. Una condición suficiente para resolverlo es que

$$m' \geq \binom{n''+1}{2} + n''$$

es decir:

$$\frac{1}{12}(n^4 - n^2) \geq \binom{\frac{n(n+1)}{2} + n - m + 1}{2} + \frac{n(n+1)}{2} + n - m.$$

Asintóticamente esto no es cierto si $m = O(n)$, pero se ha supuesto que $m = \epsilon n^2$. Luego asintóticamente, fijándonos en la parte de n^4, se obtiene que si

$$0 \leq -\epsilon^2 + \epsilon - \frac{1}{12},$$

es decir, $\epsilon \geq 0{,}09175$, entonces con el método de Relinealización se puede resolver nuestro sistema inicial en tiempo polinomial. Con el método de Linealización se pedía que asintóticamente, si $m = \epsilon n^2$, la constante ϵ fuese mayor o igual que $\frac{1}{2}$. En cambio con el método de Relinealización solo hace falta que sea mayor o igual que $\frac{1}{10}$. En caso de no poder resolver nuestro sistema con la Relinealización de grado 4 se puede volver a aplicar el método para introducir nuevas ecuaciones que ayudasen a resolver el sistema, aunque esto sería una Relinealización de grado 8.

2.2. Algoritmo XL

Ahora se introducirá el Algoritmo XL (eXtended Linearization) [CKPS00], al igual que la Relinealización, es un algoritmo de resolución de sistemas de ecuaciones no lineales en varias variables pensado para sistemas sobredefinidos. La idea del funcionamiento es más sencilla que en la Relinealización, pero su análisis es más complejo. Básicamente, la idea es multiplicar todas las ecuaciones de nuestro sistema por monomios de hasta un cierto grado d. Obviamente este proceso preserva las soluciones iniciales, para luego a través de la Linealización poder resolver nuestro sistema. El algoritmo se basa en que para un cierto grado d, suficientemente grande, la dimensión del espacio generado por nuestras ecuaciones sería muy similar al número total de monomios; por tanto, en dicha situación, se podría aplicar la Linealización para resolver el sistema.

Se presentará cómo funciona el Algoritmo XL. Hay distintas maneras de plantearlo, en función de si nuestros polinomios iniciales son homogéneos o no. Partimos de un sistema no homogéneo de m ecuaciones cuadráticas y n variables, y se trabajará con el sistema homogéneo asociado, el cual se ha obtenido introduciendo la variable x_0. Sea

$$P = \{p_i : 1 \leq i \leq m\}$$

el conjunto de ecuaciones de nuestro sistema. Por comodidad, se harán las siguientes identificaciones:

- Sea $\alpha \in \mathbb{N}^n$, entonces $x^\alpha = x_0^{\alpha_0} \cdot \ldots \cdot x_n^{\alpha_n}$.
- El conjunto de todos los monomios de grado $k \in \mathbb{N}$ lo denotamos por x^k.
- El conjunto de todos los productos de elementos de x^k y P lo denotamos por

$$x^k P = \left\{ x^\alpha p_i : x^\alpha \in x^k \text{ y } p_i \in P \right\}.$$

- $I_D = \bigcup_{k \leq D-2} x^k P$.

La manera de proceder del algoritmo XL, para un grado D fijo, sería la siguiente:

1. Se genera el conjunto I_D, multiplicando cada p_i por cada monomio de grado menor o igual que $D-2$ (como luego se verá solo hará falta considerar $x^{D-2}P$ en vez de I_D).

2. Se linealiza el sistema, considerando cada monomio de grado menor o igual que D como una nueva variable y_i. Ahora I_D es un conjunto de formas lineales. Para cada variable x_j se considera un orden en las $y_i's$, de manera que las $D+1$ variables $y_{i_1}, \ldots, y_{i_{D+1}}$, asociadas a los monomios en donde solo aparecen la x_j y x_0, estén al final. Para cada orden descrito anteriormente, se aplica Gauss al conjunto de ecuaciones en I_D, eliminando las variables de menor a mayor según el orden fijado.

3. Para cada orden descrito anteriormente, para el cual al aplicar Gauss aparezca alguna ecuación que solo dependa de las variables $y_{i_1}, \ldots, y_{i_{D+1}}$, se resolverá dicha ecuación teniendo en cuenta que $x_0 = 1$. Dichas ecuaciones se pueden resolver rápidamente en el cuerpo utilizando el Algoritmo de Berlekamp.

4. Finalmente, introducimos el valor de todas las variables x_j, halladas en el paso 3, en nuestro sistema.

Por tanto, para resolver un sistema se empieza con $D = 1$ y se irá aumentando D hasta hallar el valor de algún x_i. Una vez hallado, introducimos su valor en el sistema, se vuelve a $D = 1$ y se repetiría el proceso.

Como se verá, es más fácil trabajar con polinomios homogéneos, por eso se ha homogeneizado el sistema antes de empezar. Se verá una condición suficiente para que el XL encuentre una solución de nuestro sistema. Todo elemento de I_D se puede puede ver como un elemento del espacio vectorial V generado por el conjunto

$$B = \bigcup_{k \leq D} x^k.$$

Esto quiere decir que las coordenadas de los elementos de I_D respecto de la base B serán los coeficientes de sus términos, ordenados según el orden que se dé a los elementos de B. Por lo anterior se puede pensar en la matriz $\Pi(D)$, cuyas filas son las coordenadas de los elementos de I_D, y en las matrices $\pi(k)$ correspondientes a las coordenadas de los elementos de $x^k P$. La matriz $\Pi(D)$ se podrá poner como:

$$\Pi(D) = \begin{pmatrix} \pi(0) & 0 & \ldots & 0 & 0 & 0 \\ 0 & \pi(1) & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \pi(D) \end{pmatrix}$$

La idea, en el caso homogéneo, es encontrar a través de la Eliminación Gaussiana una

ecuación que solo dependa de x_0 y x_i. Si para un cierto $D-1$ no se ha encontrado dicha ecuación, entonces no se obtiene ningún beneficio al considerar los bloques $\pi(0),\ldots,\pi(D-1)$ en la matriz $\Pi(D)$; ya que al hacer reducciones en dichos bloques no aporta ninguna ecuación y, además, no modifica el bloque $\pi(D)$. En el caso homogéneo, solo hace falta trabajar con la matriz $\pi(D)$ y no con la matriz $\Pi(D)$. En el caso no homogéneo la matriz $\Pi(D)$ no sería una matriz por bloques y, por tanto, habría que considerar en cada etapa todas las matrices $\pi(k)$. Por tanto, en el caso homogéneo solo se tiene que trabajar con los elementos de $x^{D-2}P$ y no con los elementos de I_D.

Partimos de un sistema homogéneo de m ecuaciones y $n+1$ incógnitas, por tanto una condición suficiente para que el algoritmo XL resuelva nuestro sistema es que para un cierto D se tenga que

$$n' - rang(\pi(D)) \leq D, \tag{2.2.1}$$

siendo $n' = \left|x^D\right| = \binom{n+D}{D}$. Ya que para cada orden descrito en el paso 2, al aplicar Gauss siempre se obtendrá una ecuación que solo dependa de las últimas $D+1$ variables. Porque en el peor de los casos, la matriz $\pi(D)$ tendrá la siguiente forma:

$$\begin{array}{c|cccccc} & y_1 & \cdots & y_{n'-(D+1)} & y_{n'-D} & \cdots & y_{n'} \\ f_1 & & & & * & * & * \\ \vdots & & I_{n'-(D+1)} & & * & * & * \\ f_{n'-(D+1)} & & & & * & * & * \\ f_{n'-D} & 0 & \ldots & 0 & a_0 & \ldots & a_d \end{array}$$

con los a_i no todos nulos, obteniendo así la ecuación

$$\begin{aligned} a_0 y_{n'-D} + a_1 y_{n'-D+1} \cdots + a_d y_{n'} &= a_0 x_0^d + a_1 x_0^{d-1} x_1 + \cdots + a_d x_i^d \\ &= a_0 + a_1 x_1 + \cdots + a_d x_i^d. \end{aligned}$$

Antes de analizar el algoritmo XL, veamos porqué se puede esperar que exista un cierto D para el cual el XL resuelva el sistema.

Teorema 2.1 (Lazard)**.** *Sea P un conjunto de polinomios homogéneos que forman un sistema $\mathcal{MQ}$, entonces existe un D para el cual la Eliminación Gaussiana de $\pi(d)$, con $d=1,\ldots,D$, computa una Base de Groebner del ideal generado por P.*

Demostración. Véase [Laz83]. □

Por tanto, eligiendo un orden monomial, como los descritos anteriormente, y aplicando inducción, se tiene garantizada la resolución del sistema gracias al XL. Este teorema también muestra la relación que hay entre el cálculo de las Bases de Groebner y el XL, como luego se verá tienen bastantes similitudes. Hay que recordar que la condición 2.2.1 es una condición suficiente, por tanto, que el algoritmo termine no quiere decir que se cumpla dicha condición.

Análisis experimental

Para analizar la complejidad del XL, lo más importante es saber para un D fijo la dimensión del espacio vectorial generado por $x^{D-2}P$ o, lo que es lo mismo, saber el rango de $\pi(D)$. Una vez se sepa este dato, solo se necesita estimar un posible D para el cual el sistema pueda ser resuelto.

La siguiente tabla muestra los resultados de los test experimentales [Tho13], sobre el cuerpo $\mathbb{F}_{127}$, en la que se aprecia el valor que debería tener D, en función de $n - m$, para poder resolver nuestro sistema.

$n-m$	0	1	$c \geq 2$
D	$O(2^n)$	$O(n)$	$O(\sqrt{n})$

Cuadro 2.1: Valores de D en función de $n - m$.

Cuando $n - m \geq 2$ se tiene que $D \approx \sqrt{n}$, luego para hallar la solución de nuestro sistema se tiene que resolver mediante Gauss un sistema de

$$\binom{n+D-1}{D} \approx \frac{n^D}{D!} \approx \frac{n^{\sqrt{n}}}{\sqrt{n}!}$$

variables, por tanto en un tiempo de

$$O\left(\left(\frac{n^{\sqrt{n}}}{\sqrt{n}!}\right)^{\omega}\right) \approx O\left(e^{\omega\sqrt{n}\left(\frac{\ln n}{2}+1\right)}\right),$$

siendo $\omega = 2{,}3766$ la constante del método de Gauss, se puede resolver nuestro sistema. Este tiempo es subexponencial, aunque esto es una estimación basándonos en hechos experimentales. Como luego se verá, el análisis es mucho más complejo, de hecho aún no se sabe

en el caso general cuanto ha de valer D en función de m y n, ya que entran demasiados parámetros en juego como el valor $n-m$, el tamaño del cuerpo o la estructura particular de los sistemas.

Como ya se ha comentado, este experimento se efectuó sobre $\mathbb{F}_{127}$. En este caso las ecuaciones del cuerpo

$$x_i^q - x_i = 0$$

no han servido de ayuda, ya que D ha sido menor que 127 en las operaciones intermedias. En cambio, si estos tests se hacen sobre $\mathbb{F}_2$ las cosas cambian radicalmente, ya que dichas ecuaciones se pueden introducir. En el caso de $m = n$ y de trabajar sobre $\mathbb{F}_2$ se tiene que $D \ll 2^n$ y, además, no se aprecia tanto el cambio radical de D en función de $n-m$. Este tipo de test para ver la efectividad del algoritmo XL se puede ver en los artículos [CP03, CKPS00].

Análisis y complejidad del XL en el caso homogéneo

Ahora nos centraremos en el caso donde el sistema $\mathcal{MQ}$ venga dado por polinomios homogéneos cero dimensionales. Estos sistemas son distintos de los sistemas que aparecen en los MPKC habituales, pero estudiándolos se puede entender cuál es la complejidad del XL. Esto es lo más cerca que se puede estar de ver la complejidad del XL, ya que aún no se sabe en el caso general para que valor D el XL puede encontrar una solución, o cuál es el número de ecuaciones linealmente independientes que genera el XL para un D dado. Como ya se ha comentado, en un análisis más detallado interesan las dependencias lineales de la matriz $\pi(D)$. En el artículo original del XL [CKPS00], los autores esperan que casi todas las ecuaciones de $\pi(D)$ sean linealmente independientes, pero como se ha comprobado experimentalmente, esta afirmación no es correcta. Para ver esto, supongamos que encontramos un D para el cual el número de monomios es del orden del $rang(\pi(D))$ y, por tanto, el XL puede resolver el sistema, ¿Que relación hay entre el número de ecuaciones creadas y el número de ecuaciones independientes? Aproximadamente se tiene que

$$\frac{n^{\text{o}} \text{ ecuaciones indep}}{n^{\text{o}} \text{ ecuaciones}} \approx \frac{(n+D)(n+D-1)}{D(D-1)m}. \tag{2.2.2}$$

Por tanto, en determinadas circunstancias, no se puede asegurar que la mayoría de las ecuaciones sean linealmente independientes. Por ejemplo, en el caso de que $D \gg n$ se tendrá que el número de ecuaciones es m veces mayor que el número ecuaciones independientes.

En general, no se puede hacer un análisis del ratio de ecuaciones linealmente independientes

generadas por el XL en función del número de ecuaciones linealmente independientes en P, ya que también depende de la estructura de los polinomios en P. Por ejemplo, si inicialmente se tiene que

$$p_1 = x_1(x_2 + x_3), p_2 = x_4(x_2 + x_3) \in P,$$

aunque sean linealmente independientes, cuando se analiza $\pi(D)$ se tiene que bastantes ecuaciones se repiten, ya que dichas ecuaciones se pueden poner como h_1p_1 o h_2p_2, con $h_1, h_2 \in \mathbb{F}_q[x_1, \ldots, x_n]$. Por tanto, si nuestras ecuaciones iniciales son independientes pero tienen una cierta relación, como por ejemplo que $\gcd(p_1, p_2) \neq 1$, producen ecuaciones iguales. Para el análisis supondremos que este tipo de casos no se dan en los sistemas aleatorios $\mathcal{MQ}$ homogéneos, ya que estamos interesados en el estudio del XL aplicado a sistemas generales sin fijarnos en las posibles estructuras particulares de los mismos. Dichos sistemas $\mathcal{MQ}$ aleatorios se caracterizaran, con alguna suposición extra, con sucesiones regulares o semi-regulares.

Definición 2.2. Una sucesión de m polinomios homogéneos $\{p_1, \ldots, p_m\} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ se dice regular, si para todo $i = 1, \ldots, m$ el polinomio p_i no es un divisor de cero del anillo cociente $\mathbb{F}_q[x_1, \ldots, x_n]/\langle p_1, \ldots, p_{i-1}\rangle$, es decir, si existe un polinomio g tal que $gp_i \in \langle p_1, \ldots, p_{i-1}\rangle$ entonces $g \in \langle p_1, \ldots, p_{i-1}\rangle$.

Se sabe [Die04] que las sucesiones regulares de polinomios reflejan las propiedades de los sistemas homogéneos aleatorios cuando $m \leq n$ y la característica del cuerpo es 0. Esto quiere decir que, en característica 0, los problemas que se han mostrado antes no deberían darse, ya que los sistemas homogéneos aleatorios vienen dados por sucesiones regulares. Este resultado no se sabe para cuerpos finitos, pero se cree cierto, por tanto se puede conjeturar el siguiente resultado.

Conjetura 2.3. *Para casi todo sistema P, que sea $\mathcal{MQ}$ homogéneo con $m \leq n$, se verifica que P forma una sucesión regular.*

Por la última definición, esto quiere decir que la mayoría de los sistemas $\mathcal{MQ}$ homogéneos vienen dados por polinomios sin una estructura interna, es decir, sin relaciones distintas de las triviales. Para ilustrarlo piénsese en el conjunto $P = \{p_1, p_2\}$, el cual forma un sistema $\mathcal{MQ}$ homogéneo. Lo normal sería que no tuviesen un factor común, de hecho si fuesen aleatorios y el cuerpo tuviese característica 0 la medida del conjunto de pares $\{p_1, p_2\}$ con factores en común sería 0. Entonces, lo normal sería que un sistema $\mathcal{MQ}$ homogéneo formado por dos polinomios viniese dado por dos polinomios coprimos. ¿Que relación hay entre ser coprimo y formar una sucesión regular? La respuesta es la esperada.

Proposición 2.4. *Dos polinomios $p_1, p_2 \in \mathbb{F}_q[x_1, \ldots, x_n]$ forman una sucesión regular si, y solo si, son coprimos.*

Demostración. $\Longrightarrow$ Si comparten algún factor en común, entonces no pueden formar una sucesión regular.

$\Longleftarrow$ Supongamos que $p_1 g \in \langle p_2 \rangle$, necesariamente se tiene que $p_1 g = h p_2$. Por tanto, se obtienen las siguiente igualdades:

$$\gcd(p_2, g) = \gcd(p_2, p_1 g) = \gcd(p_2, h p_2) = p_2.$$

La primera igualdad es consecuencia de que p_1, p_2 sean coprimos. Así $p_2 \mid g$ y, por tanto, (p_1, p_2) forman una sucesión regular. □

Esto da una idea de porqué se puede conjeturar que cuando $m \leq n$ los sistemas $\mathcal{MQ}$ homogéneos y aleatorios vienen dados por sucesiones regulares. Las sucesiones regulares requieren que $m \leq n$, por tanto se necesita otra noción más que cubra el caso de $m > n$.

Definición 2.5. Un ideal $I = \langle p_1, \ldots, p_m \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ se dice cero dimensional si la correspondiente variedad afín $V(I)$, sobre la clausura algebraica $\overline{\mathbb{F}}_q$, es finita.

Definición 2.6. Un ideal I se dice homogéneo si, para cada $f \in I$, todas las partes homogéneas f^k de f también están en I.

Es muy fácil ver que un ideal I generado por polinomios homogéneos es homogéneo.

Definición 2.7. Dado un ideal homogéneo I, definimos la parte de grado d como

$$I_d = \{f \in I : deg(f) = d\}.$$

Definición 2.8. El índice de regularidad de un ideal homogéneo cero dimensional I viene dado por:

$$d_{reg} := \text{mín} \left\{ d \geq 0 : dim\, I_d = \binom{n+d-1}{d} \right\}.$$

En esta situación si se tomase $D = d_{reg}$, se tiene que el XL resolvería el sistema, ya que

$$rang(\pi(D)) = \left| x^D \right|$$

Un ideal homogéneo cero dimensional I solo tiene la solución $x = (0, \ldots, 0)$. Luego los sistemas $\mathcal{MQ}$, que estén formados por polinomios cuyo ideal sea homogéneo cero dimensional, no se dan en general en la criptografía. Pero, al menos trabajar con este tipo de

ideales, da una idea de cuál es el número de ecuaciones independientes que genera el XL. Veamos ahora la definición de sucesión semi-regular.

Definición 2.9. Una sucesión de m polinomios homogéneos $\{p_1, \ldots, p_m\} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ es semi-regular [BFS04] si, para todo $i = 1, \ldots, m$ y g, tal que $deg(gp_i) < d_{reg}$ y $gp_i \in \langle p_1, \ldots, p_{i-1} \rangle$, entonces $g \in \langle p_1, \ldots, p_{i-1} \rangle$.

Más adelante volveremos a retomar la noción de regular y semi-regular. Al igual que antes, no se ha demostrado pero se cree cierta la siguiente conjetura.

Conjetura 2.10. *Para casi todo sistema P, que sea $\mathcal{MQ}$ homogéneo con $m > n$, se tiene que P forma una sucesión semi-regular.*

Resumiendo, se quiere hallar el número de ecuaciones independientes que genera el XL para un D fijo. Entonces se ha supuesto, para simplificar las cosas, que nuestros sistemas vienen dados por polinomios homogéneos cero dimensionales y, además, se supondrá que forman una sucesión semi-regular para eliminar las posibles estructuras internas del sistema. En esta situación se tiene.

Teorema 2.11 (Moh)**.** *Si P es una sucesión semi-regular de m polinomios homogéneos en n variables que forman un sistema $\mathcal{MQ}$, entonces se tiene que:*

$$rang(\pi(D)) = \sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i+1} \binom{n+D-2i-3}{n-1}.$$

Demostración. Véase [Tho13]. □

Este teorema vale para estimar el número de ecuaciones independientes que genera el XL para casi todo sistema $\mathcal{MQ}$ homogéneo. Entonces una condición suficiente para que el algoritmo XL resuelva un sistema formado por una sucesión semi-regular de polinomios es que para un cierto D se tenga que:

$$\binom{n+D-1}{D} - \sum_{0 \leq 2i \leq D} (-1)^i \binom{m}{i+1} \binom{n+D-2i-3}{n-1} \leq D,$$

con una complejidad de

$$O\left(\left(m\binom{n+D-3}{D-2}\right)^{\omega}\right),$$

donde ω es la constante del método de Gauss.

2.3. Bases de Groebner

En esta sección se expondrá brevemente la idea de Bases de Groebner y del funcionamiento del algoritmo de Buchberger. En todo momento $\mathbb{K}$ será un cuerpo. La idea de las Bases de Groebner surge al plantearse las siguientes preguntas:

1. Dado un ideal $I \subseteq \mathbb{K}[x_1, \ldots, x_n]$, ¿será finitamente generado?
2. Dado un ideal $I = \langle f_1, \ldots, f_n \rangle$ y un polinomio $f \in \mathbb{K}[x_1, \ldots, x_n]$, ¿cuándo se puede asegurar que $f \in I$?
3. Dado un sistema de ecuaciones $P = \{p_i(x_1, \ldots, x_n) = 0 : 1 \leq i \leq m\}$, ¿cómo se puede encontrar una solución?
4. Dado un conjunto de ecuaciones paramétricas $T = \{x_i = g_i(t_1, \ldots, t_s)\}$, ¿cómo se puede encontrar un conjunto de ecuaciones que solo dependan de las $x_i's$ y que definan el mismo conjunto afín que T?

Todas estas preguntas tienen respuesta gracias a las Bases de Groebner. No vamos a responderlas en este trabajo, lo único que se hará será fijar notación y ver cuál es la idea del Algoritmo de Buchberger.

Resultados previos

Supongamos que se tiene un orden monomial en $\mathbb{K}[x_1, \ldots, x_n]$, entonces se pueden definir los siguientes conceptos:

Definición 2.12. Sea $f \in \mathbb{K}[x_1, \ldots, x_n]$ tal que $f = \sum a_\alpha x^\alpha$, entonces:

- El multigrado de f es $multideg(f) = \text{máx}\{\alpha : a_\alpha \neq 0\}$.
- El coeficiente director de f es $LC(f) = a_{multideg(f)}$.
- El monomio director de f es $LM(f) = x^{multideg(f)}$.
- El término director o cabeza de f es $LT(f) = LC(f)LM(f)$.

Dado un conjunto $F \subseteq \mathbb{K}[x_1, \ldots, x_n]$, se puede definir $LT(F) = \{LT(f) : f \in F\}$. En general dado un ideal I, que esté generado por $\{f_1, \ldots, f_s\} \subset \mathbb{K}[x_1, \ldots, x_n]$, no se verifica que

$$\langle LT(f_1), \ldots, LT(f_s) \rangle = LT(I).$$

Cuando esta igualdad se verifica se dice que el conjunto $G = \{f_1, \ldots, f_s\}$ es una Base de Groebner del ideal I. El siguiente resultado muestra que en $K[x_1, \ldots, x_n]$ se tiene la noción de división.

Teorema 2.13. *Fijado un orden monomial en $\mathbb{K}[x_1, \ldots, x_n]$. Dado un elemento $f \in \mathbb{K}[x_1, \ldots, x_n]$ y conjunto ordenado $F = (f_1, \ldots, f_s)$, se verifica que f se puede escribir de la siguiente manera:*

$$f = \sum a_i f_i + r,$$

donde $a_i, r \in \mathbb{K}[x_1, \ldots, x_n]$, $multideg(f) \geq multideg(a_i f_i)$ y, además, o $r = 0$ o ningún termino de r es divisible por ningún elemento de $LT(F)$.

Demostración. [CLO07]. □

Al elemento r lo llamamos resto y lo denotamos por $r = \overline{f}^F$. Es fácil ver que, si F es una Base de Groebner de I y $f \in I$, entonces $\overline{f}^F = 0$. Veamos finalmente cuál es el Algoritmo de Buchberger.

Algoritmo de Buchberger

Definición 2.14. Sean $f, g \in \mathbb{K}[x_1, \ldots, x_n]$ y sea $x^\gamma = lcm(LM(f), LM(g))$. Definimos el S-polinomio de f y g como:

$$S(f, g) = \frac{x^\gamma}{LT(f)} f - \frac{x^\gamma}{LT(g)} g.$$

Teorema 2.15 (Criterio de Buchberger)**.** *Sea $I \subset \mathbb{K}[x_1, \ldots, x_n]$ un ideal. El conjunto $F = (f_1, \ldots, f_s)$ es una Base de Groebner del ideal I si, y solo si, $\overline{S(f_i, f_j)}^F = 0 \ \forall i \neq j$.*

Demostración. [CLO07]. □

Algoritmo 2.16 (Buchberger)**.** *Dado un ideal $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$ no nulo, para calcular una Base de Groebner de I, se hará lo siguiente:*

`Input:` $F = \{f_1, \ldots, f_s\}$
`Output: Una Base de Groebner` G `de` I

$G := F$
`REPEAT`

$G\prime := G$
FOR $\{f, g\} \subseteq G'$ AND $f \neq g$
$S := \overline{S(p,q)}^{G'}$
IF $S \neq 0$
$G := G \bigcup \{S\}$
UNTIL $G = G\prime$

Evidentemente, este algoritmo termina en un número finito de pasos, ya que si G_i es la base parcial del paso i-ésimo se tiene que:

$$G_0 \subsetneq G_1 \dots G_{n-1} \subsetneq G_n,$$

luego por ser $K[x_1, \dots, x_n]$ Noetheriano se tiene lo deseado. El algoritmo se puede dividir en dos partes, la primera es la selección de todas las parejas posibles, cuyas reducciones de sus S-polinomios puedan formar parte de la Base de Groebner, y la segunda parte es el cálculo y reducción de los S-polinomios.

Algoritmo de Buchberger Homogéneo

Existe otro algoritmo para ideales homogéneos con un criterio distinto a la hora de seleccionar las posibles parejas, si nos damos cuenta en el algoritmo anterior se seleccionaban todas las posibles parejas. En esta subsección se verá cómo es el funcionamiento de este algoritmo.

La siguiente proposición muestra que el Algoritmo de Buchberger funciona bien para ideales homogéneos.

Proposición 2.17. *Sea $I = \langle f_1, \dots, f_s \rangle \subset \mathbb{K}[x_1, \dots, x_n]$ un ideal generado por polinomios homogéneos. El Algoritmo de Buchberger aplicado a $F = \langle f_1, \dots, f_s \rangle$ devuelve una Base de Groebner de I, cuyos polinomios son homogéneos.*

Demostración. En la primera etapa se tiene que G_0, la base parcial, está formada por polinomios homogéneos. Es fácil ver que, si f, g son homogéneos, entonces $S(f, g)$ también será homogéneo. Además, como G está formada por polinomios homogéneos, se tiene que $\overline{S(f,g)}^{G}$ será homogéneo. Luego en la primera etapa o bien se ha añadido polinomios homogéneos o bien no se han añadido ningún elemento, por tanto aplicando inducción, se tiene lo deseado. □

El siguiente algoritmo muestra una manera distinta de seleccionar las posibles parejas, como luego se verá esta selección da lugar a buenas propiedades en las bases parciales calculadas durante el algoritmo.

Algoritmo 2.18 (Algoritmo de Buchberger Homogéneo)**.** *Sea* $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$ *un ideal generado por polinomios homogéneos. Para calcular una Base de Groebner de* I*, se hará lo siguiente:*

Input: $F = \{f_1, \ldots, f_s\}$
Output: Una Base de Groebner $G =$
$(g_1, \ldots, g_l)$ de I, que verifica $totaldeg(g_i) \leq totaldeg(g_{i+1})$

$B := \{\}$
$G := ()$
$s' := 0$
REPEAT
  $d_1 := \text{mín}\{totaldeg(f) : f \in F\}$
  $d_2 := \text{mín}\{totaldeg(LCM(LT(g_i), LT(g_j))) : (i,j) \in B\}$
  $d := \text{mín}\{d_1, d_2\}$
  $B_d := \{(i,j) \in B : totaldeg(LCM(LT(g_i), LT(g_j))) = d\}$
  $B := B \setminus B_d$
  $F_d := \{f \in F : totaldeg(f) = d\}$
  $F := F \setminus F_d$
  REPEAT
    IF $B_d = \{\}$
      Elegir $f \in F_d$
      $F_d := F_d \setminus \{f\}$
    ELSE
      Elegir $(i,j) \in B_d$
      $B_d := B_d \setminus \{(i,j)\}$
      $f := S(g_i, g_j)$
    IF $\overline{f}^G \neq 0$
      $s' := s' + 1$
      $g_{s'} := \overline{f}^G$
      $G := G \cup \{g_{s'}\}$
      $B := B \cup \{(1, s'), \ldots, (s-1, s)\}$

UNTIL $B_d = \{\}$ AND $F_d = \{\}$

UNTIL $B = \{\}$ AND $F = \{\}$

La demostración del funcionamiento del algoritmo y la descripción detallada se puede ver en [KR05]. La idea de este algoritmo es similar al algoritmo anterior, conceptualmente los dos se basan en el Criterio de Buchberger. Hay partes del algoritmo que son un poco innecesarias, y que solo se hacen para que se cumpla que $totaldeg(g_i) \leq totaldeg(g_{i+1})$. Por comodidad se puede pensar que inicialmente $G = F$, ya que la condición anterior no tendría porqué preocuparnos. La diferencia de este algoritmo es que en cada etapa, en vez de seleccionar todas las posibles parejas, selecciona solo las parejas cuyo S-polinomio tenga un cierto grado. Hay que recordar que en todo momento durante la ejecución del algoritmo se está trabajando con polinomios homogéneos. En pocas palabras se puede decir que es como el Algoritmo de Buchberger pero grado por grado.

Hay varias cuestiones que surgen con este algoritmo. Por ejemplo, si se está en una etapa considerando parejas cuyo S-polinomio tenga grado d e introducimos un $g_{s'}$ en la base, todas las parejas (i, s') cuyo S-polinomio tenga grado menor que d no se contemplarán en futuras etapas, es decir, no se comprueba si

$$\overline{S(g_i, g_{s'})}^{G} = 0.$$

Como luego se verá no es necesario contemplar dichas parejas. Este tipo de criterio en el que solo se seleccionan las parejas cuyo S-polinomio tenga cierto grado, y en cada etapa el grado aumente, se llama criterio normal de selección. Veamos ahora algunas propiedades que tienen las bases parciales G_i durante las etapas del algoritmo.

Sea

$$I_{\leq d} = \{f \in I : totaldeg(f) \leq d\},$$

como se verá, el Algoritmo de Buchberger Homogéneo es capaz de calcular un conjunto que tiene las mismas propiedades que una Base de Groebner restringida a los elementos de $I_{\leq d}$.

Definición 2.19. Sea G una Base de Groebner calculada con el Algoritmo de Buchberger Homogéneo para un ideal I. Se llama la Base de Groebner d-truncada del ideal I al conjunto

$$G_{\leq d} = \{g \in G : totaldeg(g) \leq d\}.$$

Proposición 2.20. *Sea G un conjunto no vacío de polinomios homogéneos que genera un ideal I. Las siguientes condiciones son equivalentes:*

1. $G_{\leq d}$ es una Base de Groebner d-truncada de I.
2. Para todo elemento homogéneo $f \in I_{\leq d}$ se verifica que:

$$\exists g \in G_{\leq d} : LT(g) \mid LT(f).$$

Demostración. [KR05]. □

El siguiente teorema muestra que, si se interrumpiese el Algoritmo de Buchberger Homogéneo después de haber terminado la etapa para un grado d, la base parcial G sería justamente el conjunto $G_{\leq d}$. Como ya se ha comentado antes, en etapas en donde se trabaja con un grado d no se añade a la base parcial G ningún elemento con grado total menor que d. Estas propiedades surgen al utilizar el criterio de selección normal. El siguiente teorema es uno de los más importantes en este capítulo.

Teorema 2.21. *Sea $I = \langle f_1, \ldots, f_m \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$ un ideal. Sea G una Base de Groebner de I calculada con el criterio de selección normal (las parejas críticas se seleccionan de menor a mayor grado). Si durante la etapa del algoritmo, donde se han seleccionado las parejas críticas de grado d, un S-polinomio tiene grado total d o durante el proceso de reducción tiene grado total menor que d, entonces el S-polinomio será reducido a cero. Como consecuencia:*

1. Siempre que en el paso de reducción el grado total decrezca, la reducción puede pararse, ya que al final dicho elemento sería reducido al cero.
2. Los elementos nuevos que se vayan añadiendo a la base aparecen en orden creciente.

Demostración. [Tra96]. □

Lo que viene a decir este teorema es que cuando se está en una etapa con grado d no hace falta considerar S-polinomios con grado menor, ya que su reducción respecto de la base parcial sería 0. Con este teorema nos damos cuenta de que el Algoritmo de Buchberger Homogéneo funciona correctamente y que, además, el criterio de selección normal también puede aplicarse a ideales no necesariamente homogéneos. De hecho, sería recomendable utilizar este criterio ya que el número de operaciones se reduce considerablemente.

2.4. Algoritmo F4

En esta sección se presenta el algoritmo F4 [Fau99]para calcular Bases de Groebner de manera más eficiente que el Algoritmo de Buchberger, con complejidad $O(2^{3n})$ en la mayoría de los casos, donde n es el número de incógnitas. Cuando realizamos el Algoritmo de Buchberger se buscan todas las posibles parejas y se calcula los S-polinomios asociados a dichas parejas, una vez hallados se calculan los restos de los S-polinomios respecto de una base parcial. Este cálculo se hace a través de la división usual considerando un orden monomial. La idea del F4 es seleccionar unas determinadas parejas, por el criterio de selección normal y luego hacer el cálculo de los restos a través de álgebra lineal. La importancia de este algoritmo es que en la etapa del cálculo de los restos de los S-polinomios, en vez de hacerlos uno por uno, se reducirán todos a la vez por álgebra lineal. Nuestro objetivo será mostrar el funcionamiento del Algoritmo F4, así como los resultados que hacen posible su eficacia. Lo primero será ver las relaciones que hay entre el álgebra lineal y las Bases de Groebner.

Resultados previos

Lo primero que se hará será fijar ideas y notación.

Definición 2.22. Sea $F = \{f, \ldots, f_m\} \subset \mathbb{K}[x_1, \ldots, x_n]$, un conjunto de polinomios. Denotamos por $T(F)$ al conjunto de todos los monomios distintos que aparecen en los polinomios de F y por s a su cardinal.

Dado un orden monomial en $\mathbb{K}[x_1, \ldots, x_n]$, se tiene que todo polinomio de F se puede escribir como:

$$f = \sum_{i=1}^{s} c_i x^{\alpha_i},$$

con $c_i \in \mathbb{K}$ y $x^{\alpha_i} < x^{\alpha_{i+1}}$ respecto del orden monomial fijado. Por tanto, se puede definir una aplicación

$$\psi : L[T(F)] \longrightarrow \mathbb{K}^s$$

que asocie a cada polinomio sus coordenadas, donde $L[T(F)]$ es el espacio vectorial generado por $T(F)$ y $\psi(f) = (c_1, \ldots, c_s)$. Por tanto, se puede identificar un polinomio como un vector asociado o viceversa. Finalmente, se tiene la siguiente matriz, cuyas filas son las

coordenadas de los polinomios de F,

$$\Psi(F) = \begin{pmatrix} \psi(f_1) \\ \vdots \\ \psi(f_m) \end{pmatrix}.$$

Las siguientes definiciones serán útiles en la descripción del Algoritmo F4.

Definición 2.23. Sea F un conjunto de polinomios. Definimos:

- El conjunto de polinomios asociados a las filas de la matriz escalonada por filas de $\Psi(F)$, se denotará por $\widetilde{F}$.
- Sea $\widetilde{F}^+ = \left\{ g \in \widetilde{F} : LT(g) \notin LT(F) \right\}$ y $\widetilde{F}^- = \widetilde{F} \setminus \widetilde{F^+}$.

Estamos interesados en una base triangular B de F, es decir, $\forall \psi(f), \psi(g) \in B$ se ha de verificar que $LM(f) \neq LM(g)$. Al trabajar con los elementos de B, y no con los de F, se desecharán los polinomios que son innecesarios en cálculos intermedios. Para ello se tiene el siguiente teorema.

Teorema 2.24. *Sea F un conjunto finito de polinomios y $s = |T(F)|$. Para cualquier subconjunto $H \subseteq F$ tal que $|H| = |LT(F)|$ y $LT(H) = LT(F)$, se tiene que*

$$B = \left\{ \psi(g) : g \in \widetilde{F}^+ \cup H \right\}$$

forman una base triangular del espacio vectorial $L[\{\psi(f) : f \in F\}]$.

Demostración. Todos los elementos de $\widetilde{F}^+ \cup H$ tienen distinto monomio director, luego el conjunto B es un sistema linealmente independiente y triangular. Sea $r = \dim(L[\{\psi(f) : f \in F\}])$, se tiene que $r = rang(\Psi(F))$ y, además, como la matriz $\Psi(F)$ es triangular se tiene que

$$r = rang(\Psi(F)) = \left| LT(\widetilde{F}) \right|$$

Como $LT(\widetilde{F}^+ \cup H) = LT(\widetilde{F})$, se tiene que $r = LT(\widetilde{F}^+ \cup H) = |B|$. Finalmente como $B \subseteq L[\{\psi(f) : f \in F\}]$ se llega a que B genera $L[\{\psi(f) : f \in F\}]$. □

La idea principal del F4 es hacer la reducción de los S-polinomios a través de la Eliminación Gaussiana. La idea es seleccionar un conjunto de posibles candidatos y, a partir de Gauss,

decidir cuáles se incluirán en la base. Supongamos que se ha elegido el criterio de selección normal y la base parcial es G. Entonces se define el conjunto de parejas críticas de grado d de la siguiente manera:

$$B_d = \{(b_1, b_2) : b_i \in G \text{ con } totaldeg(LCM(LT(b_1), LT(b_2))) = d \text{ y } b_1 \neq b_2\}$$

El siguiente conjunto está formado por los elementos con los que se trabajarán para determinar qué se deberá introducir en la base parcial.

$$L_d = \bigcup_{(b_1,b_2)\in B_d} \left\{ \frac{LCM(LT(b_1), LT(b_2))}{LT(b_1)} b_1, \frac{LCM(LT(b_1), LT(b_2))}{LT(b_2)} b_2 \right\}$$

Ahora se quieren calcular los restos de L_d respecto de la base parcial G. Ya se ha comentado que este proceso se hará por Eliminación Gaussiana. Pero entonces surge un problema. Con la división habitual se pueden hacer más operaciones que solamente considerando combinaciones lineales, ya que en la división normal se considera combinaciones lineales y producto por monomios. Esto se puede solucionar gracias a los reductores.

Sea G la base parcial y F un conjunto de polinomios, supongamos que se tiene un elemento $m \in T(F) \setminus LT(F)$ de manera que existe un elemento $g \in G$ con $LT(g) \mid m$. Entonces si se hiciese la reducción, es decir, el cálculo del resto respecto de G, con la división habitual se sabe que dicho monomio podría ser eliminado multiplicando a g por un cierto elemento. Para eliminar ese termino m con operaciones lineales se tiene que considerar el elemento $g\frac{m}{LT(g)}$. Naturalmente surge la siguiente definición.

Definición 2.25. Durante el proceso de cálculo de Bases de Groebner con base parcial G, un reductor r, respecto de un conjunto de polinomios F, es un elemento de la forma

$$g\frac{m}{LT(g)},$$

con $m \in T(F) \setminus LT(F)$ y $g \in G$.

El siguiente algoritmo añade a un conjunto F todos los posibles reductores. Nótese que para poder hacer todas las operaciones posibles de la división, solo con combinaciones lineales, hay que considerar los reductores de los reductores y así sucesivamente, hasta que no haya ningún reductor más.

Algoritmo 2.26 (Cálculo de reductores)**.** *El siguiente algoritmo calcula todos los reductores de F respecto de una base intermedia G.*

```
Input: F = {f_1, ..., f_s}  y  G = {g_1, ..., g_m}
Output: F ∪ R

D := LT(F)
R := {}
WHILE T(F ∪ R) ≠ D DO
    Elegir m ∈ T(F ∪ R) \ D
    D := D ∪ {m}
    IF ∃g ∈ G tal que LT(g) | m
        R := R ∪ {g m/LT(g)}
```

Ahora se está en situación de ver la reducción utilizada en el F4. Supongamos que se está en una etapa del F4 con una base intermedia G y se ha calculado el conjunto L_d, entonces para calcular la reducción lo primero que se calcula es $F = L_d \cup R$ con el algoritmo anterior, para luego calcular $\widetilde{F}^+$. A este proceso se le denotará por $Reducción F4(L_d, G)$.

$$(L_d, G) \longrightarrow F = L_d \cup R \longrightarrow \widetilde{F}^+$$

Con este proceso lo que se hace es quedarse con los elementos que se introducirán en la base parcial G. Pero, ¿qué se está haciendo realmente? Los elementos de L_d son múltiplos de elementos de G, luego no nos interesan, en cambio lo que sí nos interesa son sus restos respecto de la base parcial G. Como lo que se quiere es calcular los restos con álgebra lineal se tienen que introducir los reductores. Una vez introducidos los reductores se calcula $\widetilde{F}^+$, dicho conjunto es justamente los restos respecto de G. Nótese que durante el cálculo de $\widetilde{F}^+$ descartamos los elementos de R, dichos elementos son múltiplos de elementos de G, luego no nos interesan. Obviamente hay que comprobar que todos estos argumentos dados son ciertos, para ello nos basamos en los siguientes resultados.

Lo primero es ver que los elementos de $\widetilde{F}^+$ contribuyen al ideal generado por los términos directores de la base parcial.

Lema 2.27. *Supongamos que se está en una etapa intermedia del algoritmo F4. Sea* $\widetilde{F}^+ = Reducción F4(L_d, G)$*, entonces* $\forall f \in \widetilde{F}^+$ *se verifica que* $LT(f) \notin \langle LT(G) \rangle$.

Demostración. Sea $f \in \widetilde{F}^+$, supongamos que $LT(f) \in \langle LT(G) \rangle$ para llegar a una contradicción. Por la hipótesis se tiene que:

$$a_\alpha x^\alpha = LT(f) = \sum a_i LT(g_i),$$

con $a_i \in \mathbb{K}[x_1, \ldots, x_n]$. Por hipótesis, necesariamente existe algún g_i con $LT(g_i)$ dividiendo a $a_\alpha x^\alpha$. Entonces como $LT(f) \in T(\widetilde{F}^+) \subset T(F)$ y $LT(f) \notin LT(F)$, ya que $f \in \widetilde{F}^+$, durante el cálculo de reductores se tendría que haber añadido $\frac{LT(f)}{LT(g)}g$ a F. Pero entonces se tiene que:

$$LT(f) = LT\left(\frac{LT(f)}{LT(g)}g\right) \in LT(F),$$

lo cual es una contradicción ya que $f \in \widetilde{F}^+$. □

Este lema asegura que se están añadiendo elementos nuevos que contribuyen a que G sea una Base de Groebner. El siguiente lema muestra que los elementos que se están añadiendo son parte del ideal.

Lema 2.28. *Sea $\widetilde{F}^+$ como en el lema anterior. Se verifica que $\widetilde{F}^+ \subset \langle G \rangle$.*

Demostración. Los elementos de $\widetilde{F}^+$ son combinaciones lineales de elementos de L_d y R, los cuales están contenidos en $\langle G \rangle$. □

Este lema y el anterior muestran que en cada etapa se aumentará $\langle LT(G) \rangle$. Por lo que al ser $\mathbb{K}[x_1, \ldots, x_n]$ Noetheriano, el algoritmo terminará en un número finito de pasos. Solo hace falta ver que es lo que pasa con los S-polinomios para poder aplicar el Criterio de Buchberger. Ahora se aplicarán todos los pasos previos, referentes a los reductores y a las reducciones por álgebra lineal, para demostrar el siguiente resultado.

Lema 2.29. *Sea $\widetilde{F}^+$ como en el apartado anterior. Toda combinación lineal de elementos de L_d es reducida a cero respecto de $\widetilde{F}^+ \cup G$.*

Demostración. Sea $F = L_d \cup R$, entonces se tiene que:

$$\Psi(F) = \begin{pmatrix} L_d \\ R \end{pmatrix},$$

pensando en los polinomios como vectores. Sea $\Psi'(F)$ la matriz escalonada por filas de la matriz $\Psi(F)$, se tiene que $\Psi'(F)$ se puede poner como:

$$\Psi'(F) = \begin{pmatrix} \widetilde{F}^- \\ \widetilde{F}^+ \end{pmatrix}.$$

Por definición los elementos $f_i \in \widetilde{F}^-$ verifican que $LT(f_i) \in LT(G)$, luego como $T(\Psi(F)) = T(\Psi'(F))$, en el paso de añadir todos los reductores se tuvo que añadir el elemento $g_i \in G$,

con $LT(g_i) = LT(f_i)$, a R. Como la matriz $\Psi'(F)$ es escalonada por filas, si se considera la matriz $\Psi''(F)$, obtenida al sustituir la filas correspondientes a las f_i por las filas correspondientes a los g_i, se obtiene que

$$rang(\Psi(F)) = rang(\Psi'(F)) = rang(\Psi''(F)).$$

La matriz $\Psi''(F)$ tiene la siguiente forma:

$$\Psi''(F) = \begin{pmatrix} G' \\ \widetilde{F}^+ \end{pmatrix},$$

con $G' \subseteq G$. Por otra parte, ya se ha visto que las filas correspondientes a los elementos g_i están en $\Psi(F)$. Por tanto, se tiene que el espacio generado por las filas de la matriz $\Psi(F)$ es el mismo que el espacio generado por las filas de la matriz $\Psi''(F)$. Entones se ha llegado a que, si $h \in L[L_d] \subseteq L[F]$ se verifica que $h \in L[G' \cup \widetilde{F}^+]$ por tanto:

$$h = \sum_{i=1}^{k} a_i h_i, \tag{2.4.1}$$

con $a_i \in K$ y $h_i \in G' \cup \widetilde{F}^+$. Luego, el resto de h respecto de $G \cup \widetilde{F}^+$ será 0, ya que es una combinación lineal de elementos de $G' \cup \widetilde{F}^+$. □

Como consecuencia a este lema, todos los S-polinomios de grado d son combinaciones lineales de elementos de $G \cup \widetilde{F}^+$.

Observación 2.30. En el último paso de la demostración, si no se hubiera obtenido una combinación lineal, por ejemplo si los a_i fuesen polinomios, el argumento no valdría.

Este lema, junto con los dos anteriores y el Criterio de Buchberger, asegura que el Algoritmo F4 devuelve una Base de Groebner en un número finito de pasos.

Algoritmo F4

Ahora se está en condiciones de presentar el algoritmo.

Algoritmo 2.31 (F4)**.** *El algoritmo F4 devuelve una Base de Groebner, G, de un ideal I generado por $F = \{f_1, \ldots, f_m\}$ en un número finito de pasos.*

`Input:` $F = \{f_1, \ldots, f_m\}$

Output: G una Base de Groebner de $I = \langle f_1, \ldots, f_m \rangle$

$G := F$
$\widetilde{F}_0^+ := F$
$d := 0$
$B := \{(b_1, b_2) : b_i \in G, b_1 \neq b_2\}$
WHILE $B \neq \emptyset$ DO

$d := d + 1$
$B_d := \{(i,j) \in B : totaldeg(LCM(LT(g_i), LT(g_j))) = d\}$
$B := B \setminus B_d$
$L_d := \bigcup_{(b_1,b_2)\in B_d} \left\{ \frac{LCM(LT(b_1),LT(b_2))}{LT(b_1)} b_1, \frac{LCM(LT(b_1),LT(b_2))}{LT(b_2)} b_2 \right\}$
$\widetilde{F}_d^+ := reducciónF4(L_d, G)$
FOR $f \in \widetilde{F}_d^+$

$B := B \cup \{(f,g) : g \in G, LCM(LT(f), LT(g_2)) \geq d\}$
$G := G \cup \{f\}$

En este algoritmo se han seleccionado las parejas críticas B_d, según el criterio de selección normal, pero se puede utilizar otros criterios para optimizar el algoritmo. Por ejemplo en el algoritmo F5 se utiliza el criterio de Gebauer y Möller [GM88], con el que se reduce el número de parejas críticas no necesarias.

En el artículo [SK06], se muestran las similitudes entre el Algoritmo F4 y el XL. En dicho artículo se muestra cómo los polinomios involucrados en el F4 forman un subconjunto de los polinomios que se generan durante la ejecución del XL. Se cree que el XL es capaz de encontrar una Base de Groebner para un D más pequeño que el necesario por el F4, ya que el número de polinomios es mayor y, por tanto, el número de reducciones (aunque experimentalmente, las matrices involucradas en el F4 son más pequeñas, ya que los criterios de selección de parejas críticas desechan polinomios inútiles en los cálculos intermedios).

La complejidad del algoritmo no está clara, solo hay pruebas experimentales que justifican que en la mayoría de los casos el algoritmo tiene una complejidad de $O(2^{3n})$. Es un tema muy estudiado en la actualidad por las consecuencias tan grandes que tiene en el álgebra computacional.

Capítulo 3

Complejidad de las Bases de Groebner

Una de las herramientas principales para analizar sistemas MPKC son las Bases de Groebner, ya que un sistema MPKC deberá ser resistente a este tipo de ataques. Estos algoritmos muestran cuándo un sistema tiene una estructura interna, lo cual indica que el sistema no se parece al prototipo de sistema aleatorio. Es decir, si un algoritmo de cálculo de Bases de Groebner tiene una complejidad mucho menor en nuestro sistema que en los sistemas aleatorios, quiere decir que las ecuaciones del sistema verifican relaciones polinómicas. El algoritmo F4 fue capaz de romper el famoso sistema HFE [FJ03], lo cual hizo que se tomase como referencia.

Además, el estudio de sistemas de ecuaciones polinomiales tiene un gran interés, no solo por su aplicación en criptografía multivariable, sino también por su aplicación en criptografía simétrica como el AES [AA10]. Se puede transformar un cifrado AES en un sistema de ecuaciones disperso de unas 6000 ecuaciones y 1600 variables. En este apartado se estudiarán las características principales de las Bases de Groebner [DHK$^+$13, BFS04, BFSY05, BFS15, BcFSyY, BFS03] y se verá de qué depende su complejidad en determinadas situaciones. Primero se empezará por las propiedades básicas y más adelante se estudiarán propiedades que solo se verifican en determinadas ocasiones. A menos que se diga lo contrario, se trabajará con órdenes monomiales graduados.

Proposición 3.1. *Criterio del producto: Si una pareja verifica que* $\gcd(f, g) = 1$ *entonces dicha pareja no tiene porqué ser contemplada.*

Demostración. En dicha situación el S-polinomio será

$$S(f,g) = \frac{LM(f)LM(g)}{LT(f)}f - \frac{LM(f)LM(g)}{LT(g)}g.$$

Por tanto, o f o g dividen a $LT(S(f,g))$. □

En el algoritmo F4 cuando se tiene la matriz F_d antes de proceder con el escalonamiento de la matriz y obtener $\widetilde{F}_d$ se pueden hacer simplificaciones de las filas de F basándonos en los anteriores escalonamientos, i.e, si en una etapa anterior se obtiene que $f \to f'$ entonces si en la matriz F_d aparece λf se puede simplificar $\lambda f \to \lambda f'$. A veces ese tipo de reducciones no ayudan.

Teorema 3.2 (Teorema de eliminación)**.** *Sea $I \subseteq k[x_1, \ldots, x_n]$ un ideal y sea G una base de Groebner de I respecto del orden lexicográfico. Entonces para cada $0 \leq i \leq n$, el conjunto*

$$G_i = G \cap k[x_{i+1}, \ldots, x_n]$$

es una base de Groebner del i-ésimo ideal de eliminación I_i.

Demostración. [CLO07] □

Observación 3.3. Si se tiene un sistema de ecuaciones y pensamos en la variedad formada por las soluciones del sistema, lo que conseguimos con el i-ésimo ideal de eliminación es proyectar la variedad. Por tanto, si conseguimos calcular G_n obtendremos ciertos polinomios que solo dependerán de la variable x_n; con los que se puede obtener la coordenada n-ésima de la solución de nuestro sistema inicial.

Teorema 3.4. *Una base $G = (g_1, \ldots, g_t)$ es una base de Groebner para el ideal I si, y solo si, para todo elemento $S = (h_1, \ldots, h_t)$ de una base homogénea de las sicigias $S(G)$, se tiene que*

$$S \cdot G = \sum h_i g_i \longrightarrow_G 0$$

Demostración. [CLO07] □

Teorema 3.5. *Sea $G = \{g_1, \ldots, g_n\}$ un conjunto de polinomios y $S(G)$ el espacio de las sicigias de $LT(G)$. Se verifica que toda sicigia f se puede poner como combinación con coeficientes polinomiales de los S-polinomios de G, es decir:*

$$f = \sum f_i e_i = \sum h_{ij} S_{ij} \quad h_{ij} \in k[x_1, \ldots, x_n], S_{ij} = \frac{x^{\gamma_{ij}}}{LT(g_i)}e_i - \frac{x^{\gamma_{ij}}}{LT(g_j)}e_j$$

Demostración. [CLO07] □

Criterio 3.6 (Criterios para la eliminación de parejas innecesarias)**.**

- Solo se comprueban los $S(f,g)$ una sola vez, ya que si son 0 en una etapa seguirán siendo 0 en las etapas posteriores y si no son 0 se habrán añadido anteriormente a la base.
- Si $\gcd(LT(f), LT(g)) = 1 \Rightarrow S(f,g) \longrightarrow_G 0$
- Sea $S \subset \{S_{ij}\}$ una base de $S(G)$. Supongamos que $g_i, g_j, g_k \in G$ tal que

$$LT(g_k)/lcm(LT(g_i), LT(g_j)).$$

  Si $S_{ik}, S_{jk} \in S$, entonces $S - \{S_{ij}\}$ es también una base de $S(G)$.

El k-ésimo ideal de eliminación es el ideal de la variedad que contiene la proyección de la variedad sobre un subespacio. Si una ecuación f está en el ideal de eliminación quiere decir que la variedad está contenida en un producto de la forma $V(f) \times W$. Eso se tiene en el caso de un número finito de puntos, ya que siempre se puede proyectar a una recta y obtener una ecuación que se anule en los puntos proyectados, por tanto cuando trabajamos con un número finito de puntos dichas ecuaciones han de aparecer en los ideales de eliminación. Una vez obtengamos esas ecuaciones conseguiremos un número finito de posibles valores para una cierta coordenada, para luego seguir con un proceso recursivo.

3.1. Índice de Regularidad

En esta sección retomamos con más detalle las sucesiones regulares y semi-regulares, ya que con ellas se podrá entender mejor cuál es la complejidad del F4 en determinados casos.

Sea $S = \mathbb{K}[x_1, \ldots, x_n]$, $I^i = (f_1, \ldots, f_i)$, I_k formas de grado k de I (polinomios homogéneos de grado k), $A^i = S/I^i$, $I^i_k = \left(I^i\right)_k$ y $A^i_k = \left(A^i\right)_k$. Siempre se trabajará con órdenes graduados, como el grlex (Orden lexicográfico graduado). Para estudiar la complejidad de las Bases de Groebner uno de los datos más importantes a tener en cuenta son los grados de los polinomios involucrados en dicha base o durante la ejecución del algoritmo. Estamos interesados en el estudio de sistemas sobredefinidos ya que se necesita que la aplicación de cifrado sea invertible y, por lo tanto, inyectiva. Como luego se verá, nos podemos restringir al caso de ideales cero dimensionales. Nuestro objetivo será determinar una cota para el

grado máximo de los polinomios que forman una Base de Groebner de un ideal dado y ver que dicha cota es válida para casi todo sistema. Gracias a esa cota se puede determinar cuál es la complejidad del F4. Empezamos trabajando con polinomios homogéneos, ya que con ello se conseguirá simplificar los argumentos; más tarde se justificará el caso no homogéneo.

Definición 3.7. Sea $I = (f_1, \ldots, f_m)$ un ideal homogéneo de $S = K[x_1, \ldots, x_n]$, se dice que la sucesión $f_1, \ldots, f_m$ es regular si la aplicación multiplicar por f_i

$$(S/(f_1, \ldots, f_{i-1}))_{a-d_i} \xrightarrow{*f_i} (S/(f_1, \ldots, f_{i-1}))_a$$

es una aplicación lineal inyectiva para todo a y para todo i, donde d_i es el grado de f_i. Es decir, para todo i se verifique que f_i no es un divisor de cero del anillo $(S/(f_1, \ldots, f_{i-1})$. Si la aplicación $*f_i$ tiene rango máximo, entonces la sucesión se dirá que es semi-regular. Claramente toda sucesión regular es semi-regular.

Esta definición de regular es equivalente a la que dimos anteriormente, pero no así la definición de semi-regular, luego veremos que relación hay entre ellas. Por otra parte como luego se verá, una sucesión semi-regular es regular cuando $m \leq n$ y, además, no hay sucesiones regulares cuando $m > n$. De hecho, lo que se verá es que si una sucesión $(f_1, \ldots, f_m)$ es semi-regular entonces las sucesiones $(f_1, \ldots, f_s)$ son semi-regulares para $s \leq m$ y en particular regular cuando $s \leq min\{m, n\}$. La noción de regular formaliza la noción de un sistema de ecuaciones sin relaciones entre sí. Estamos interesados en las propiedades de los sistemas 'aleatorios', i.e, estamos interesados en las propiedades genéricas de los sistemas de ecuaciones. Lo que se quiere decir con genérico es que si se ven los sistemas de ecuaciones como elementos del espacio vectorial $\prod S_{d_i}$, entonces dicha propiedad se verifica en un abierto de Zariski

$$U \subseteq \prod S_{d_i}$$

con lo que dicha propiedad sería de esperar que se verificase para 'casi todo' sistema. La expresión 'casi todo' hace referencia a que U^c tiene medida nula si el cuerpo es infinito, ya que se puede pensar en S_{d_i} como un compacto (los polinomios son puntos de un espacio proyectivo). En el caso de que el cuerpo sea finito habría que cambiar la expresión 'casi todo' por 'un sistema genérico'. Como luego se verá la medida U^c tiende a 0 cuando el tamaño del cuerpo tiende a infinito.

En este sentido se tiene la siguiente conjetura.

Conjetura 3.8. *Si $\mathbb{K}$ es un cuerpo infinito, entonces la propiedad de ser semi-regular es genérica.*

En el caso regular y el cuerpo infinito la conjetura es cierta. Esta conjetura no se ha demostrado pero experimentalmente se ha visto que, tanto en cuerpos finitos como en infinitos, se verifica. Esto quiere decir que experimentalmente la mayoría de los sistemas vienen dados por sucesiones semi-regulares de polinomios.

Que una sucesión sea regular en particular dice que no hay relaciones con coeficientes polinomiales, lo que sería de esperar en sistemas aleatorios. También se tiene que las sucesiones regulares son aquellas que forman una variedad con intersección completa, lo cual también es esperable en sistemas aleatorios. Hay otra definición de sucesión semi-regular no equivalente dada por Faugère que ya vimos anteriormente, veamos qué relación hay entre ambas, aunque primeramente veremos unos conceptos previos.

Definición 3.9. Para un ideal homogéneo definimos el índice de regularidad del ideal como

$$d_{reg} = min\left\{d \geq 0 \mid dim\,(I_d) = \binom{n+d-1}{d}\right\}$$

Si dicho mínimo no existe, entonces definimos $d_{reg} = \infty$.

Lo primero es ver cuando se puede esperar que dicho mínimo exista.

Definición 3.10. Sea I un ideal homogéneo del anillo graduado A, la Función de Hilbert de I en el anillo A se define como:

$$HF_{A,I} = HF_I(l) = \dim\,(A/I)_l = \dim A_l - \dim I_l$$

Observación 3.11. Evidentemente la función depende del anillo A aunque normalmente no se pondrá cuando se sepa sobre qué anillo se está trabajando. En el caso de que A sea el anillo de coordenadas $\mathbb{K}[x_1, \ldots, x_n]$ se tendrá que:

$$HF_I(l) = \binom{n+d-1}{d} - \dim I_l$$

Siempre se puede calcular la función de Hilbert tomando una resolución del ideal, en ese caso la función de Hilbert quedara en función de ciertos binomios. Algunos de dichos monomios no tendrán sentido para l pequeños, pero para un l suficientemente grande todos los binomios tendrán sentido y, por tanto, la función de Hilbert vendrá expresada en un polinomio $HP(l)$, para l suficientemente grande. Evidentemente la función de Hilbert es una función no negativa con la propiedad de que si $HF_I(l) = 0$ entonces para todo $l < l'$ se tiene que $HF_I(l') = 0$.

Lema 3.12. *Sea HF_I la función de Hilbert de I en S, entonces $HF_I(l) = \dim(S/I)_l = \binom{n+d-1}{d} - \dim I_l = 0$ para $l >> 0$ si, y solo si, $V_{\overline{\mathbb{K}}}(I) = \emptyset$.*

Demostración. [Eis95, Kun12] □

El lema anterior dice justamente que dicho mínimo existe cuando $V(I) = \emptyset$ (como variedad proyectiva). En el caso de trabajar con sucesiones semi-regulares se tiene lo siguiente:

- Si $n > m$ entonces el sistema es regular y, por tanto, la variedad es intersección completa. En este caso la $\dim V(I) = n - m - 1 \geq 0$, con lo que $V(I) \neq \emptyset$ y, por tanto, $d_{reg} = \infty$.
- Si $n \leq m$ entonces en particular $(f_1, \ldots, f_n)$ es regular con $\dim V(f_1, \ldots, f_n) = -1$, esto lleva necesariamente a que $V(I) = \emptyset$ y, por tanto, $d_{reg} < \infty$.

El polinomio de Hilbert o es constantemente cero o siempre positivo, ya que si en algún momento la función de Hilbert toca el 0 ha de quedarse en él. Por tanto, si el índice de regularidad existe para un cierto ideal, quiere decir que su polinomio de Hilbert necesariamente es cero y, por tanto, $V(I) = \{0\}$ visto como variedad afín. También se puede ver que si el índice de regularidad existe y los polinomios son homogéneos necesariamente $V(I) = \{0\}$ ya que como las dimensiones del número de monomios de grado d_{reg} y I_{dreg} son las mismas, utilizando linealización se llega a que $x_i^{dreg} = 0$.

Otra de las razones de pedir que los polinomios sean homogéneos es que facilitan el estudio de la complejidad. Para ello se utiliza la siguiente proposición.

Proposición 3.13. *Sea I un ideal generado por m polinomios homogéneos con $n \leq m$. Entonces la base de Groebner G del ideal verifica que:*

$$\max\{\deg(g) \mid g \in G\} \leq d_{reg}$$

donde d_{reg} es el índice de regularidad del ideal I.

Demostración. Supongamos, sin pérdida de generalidad, que G es una base de Groebner reducida. Supongamos lo contrario, i.e, $\exists g \in G$ tal que $\deg g > d_{reg}$. Se tiene que $LT(g)$ es divisible por un monomio $m \in S_{d_{reg}} = I_{d_{reg}}$; pero dicho monomio está en el $LT(I)$ y, por tanto, existirá un elemento $g' \in G$ de manera que

$$LT(g')/m/LT(f)$$

lo cual es contradictorio ya que $\deg g' < \deg g$ y G es reducida. De hecho se tiene que todo monomio de grado d_{reg} será divisible por $LT(f)$ para cierto f en G. □

Como luego se verá, el hecho de conocer el índice de regularidad de un ideal proporciona una manera de conocer la complejidad de calcular una base de Groebner de dicho ideal. Experimentalmente se puede ver que el d_{reg} es una buena cota para el grado máximo de los elementos de una base de Groebner [Die15]. Como se ve en la tabla 3.1 el índice de regularidad acota bastante bien el grado máximo. La tabla muestra el resultado del grado máximo de los polinomios que intervienen en una base de Groebner para un ideal de n variables y $2n$ ecuaciones sobre $\mathbb{F}_2$, tomando como muestra sistemas homogéneos de grado 2 al azar. La tabla ha sido generada con los programas que he desarrollado en SAGE, pueden verse al final. Como luego se verá, para la mayoría de los sistemas el índice de regularidad solo depende del número de variables, ecuaciones y grados de los polinomios involucrados.

Nº Variables	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
d máximo	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5	5
d_{reg}	3	3	3	4	4	4	4	4	4	4	5	5	5	5	5	5	5

Cuadro 3.1: Comparación entre el d_{reg} y el grado máximo en una base de Groebner.

En el caso no homogéneo no se puede trabajar directamente con el ideal, lo que se hará es trabajar con las formas de mayor grado de los polinomios que forman el ideal. La siguiente proposición da una cota para el grado de los polinomios de una base de Groebner.

Proposición 3.14. *Sea I un ideal y I' el ideal generado por las partes homogéneas de mayor grado de los polinomios que generan I. Si G es una base de Groebner de I se verifica que:*

$$\max\{\deg(g) \mid g \in G\} \leq d_{reg}$$

donde d_{reg} es el índice de regularidad del ideal I'.

Demostración. Sin pérdida de generalidad se puede suponer que G es una base de Groebner reducida. Supongamos lo contrario, i.e, $\exists g \in G$ tal que $\deg g > d_{reg}$. Se tiene que $LT(g)$ es divisible por un monomio $m \in S_{d_{reg}} = I'_{d_{reg}}$. Esto implica necesariamente que $m \in LT(I') \subseteq LT(I)$, siempre que el orden sea graduado. Por tanto, existe un $g' \in G$, tal que

$$LT(g')/m/LT(g)$$

y, además, con $\deg m < \deg g$, por tanto al ser G una base reducida se llega a una contradicción. □

Ahora se está en condiciones de dar la definición alternativa que da Faugère.

Definición 3.15 (Faugère)**.** Sea $I = (f_1, \ldots, f_m)$ un ideal homogéneo. Se dirá que la sucesión $(f_1, \ldots, f_m)$ es semi-regular si para todo i y todo g tal que $deg(f_i g) < d_{reg}$ y $gf_i \in (f_1, \ldots, f_{i-1})$ entonces se tiene que g está en el ideal $(f_1, \ldots, f_{i-1})$.

Si $n > m$ entonces se ve fácilmente que la definición de Faugère es la misma que la definición de regular, ya que en este caso $d_{reg} = +\infty$. Esta segunda definición dada por Faugère viene a decir que un sistema es semi-regular cuando es regular hasta un cierto grado, donde dicho grado es el máximo posible en el que se puede cumplir dicha condición.

Ahora se verá cuál es la relación entre ambas definiciones, pero antes veamos primero una definición más.

Definición 3.16. Se dice que f es semi-regular (regular) en S/I si la aplicación

$$(S/I)_{a-d} \xrightarrow{*f} (S/I)_a$$

multiplicar por f, es lineal de rango máximo (inyectiva).

Veamos qué relación hay entre la definición de semi-regular y la definición de semi-regular dada por Faugère.

Supongamos que se tiene un ideal homogéneo I que, además, verifica que S/I es Artiniano, se sabe que [PR12]:

$$S/I \text{ Artiniano } \Leftrightarrow S/I \text{ noetheriano de dimensión } 0 \Leftrightarrow V_{\overline{\mathbb{K}}}(I) \text{ es finito}$$

entonces se sabe, por las consideraciones anteriores, que existe el índice de regularidad. Se podría decir que los ideales en los que estamos interesados son aquellos tales que S/I es artiniano. Si ρ es el mayor grado para el cual S/I no es cero (índice de Castelnuovo-Mumford [Die15]), entones se tiene que $\rho = d_{reg} - 1$ y, además, se verifica la siguiente proposición.

Proposición 3.17. *Sea $f \in S$ una forma de grado mayor que ρ. Se verifica que f es semi-regular en S/I.*

Demostración. Se tiene que en la aplicación de multiplicación por f o el dominio o el codominio es 0 para todo a, lo cual conduce inmediatamente a que la aplicación tiene rango máximo. □

Definimos $I^i = (f_1, \ldots, f_i)$ y $d_{reg,i} = d_{reg}(I^i)$. Análogamente al razonamiento de la demostración anterior solo es necesario comprobar que la aplicación multiplicar por f_i sea de rango máximo para $a - d \leq \rho_i$ por tanto $a - d < d_{reg,i}$ para saber que f_{i+1} es semi regular en S/I^i. La aplicación será de rango máximo cuando sea o suprayectiva o inyectiva, por tanto se determinará cuándo la aplicación es suprayectiva.

Proposición 3.18. *Sea $f_i \in A^{i-1} = S/I^{i-1}$ una forma . Se verifica que la aplicación multiplicación por f_i es suprayectiva para el grado a si, y solo si,* $\dim A^i_a = 0$.

Demostración. Se tiene la siguiente sucesión exacta:

$$\ker(*f_i) \to A^{i-1}_{a-d_i} \xrightarrow{*f_i} A^{i-1}_a \to A^i_a \to 0$$

Si la aplicación $*f_i$ es suprayectiva en el grado a entonces $A^{i-1}_{a-d_i} f_i \supseteq A^{i-1}_a$. Por otra parte se tiene que:

$$A^i_a = \left(A^{i-1}/f_i A^{i-1}\right)_a = A^{i-1}_a / f_i A^{i-1}_{a-d_i}$$

Por lo que $*f_i$ es suprayectiva en el grado a si, y solo si, $\dim A^i_a = 0$. □

Observación 3.19. Cuando se calcula la HS_I para una sucesión semi-regular sobredefinida se obtiene, que para valores k mayores o iguales a d_{reg}, el coeficiente c_k es cero. En cambio, si se calcula dicho coeficiente con la fórmula que luego se verá, lo que se obtiene es un número negativo. Justamente la fórmula está pensada para que la aplicación sea inyectiva y el sumando $-\dim(\ker(*f_i))$ sea 0. En cambio cuando la aplicación no es inyectiva ese sumando no es cero y, por tanto, aparecen dichos números negativos.

Como ya se ha comentado antes $\dim A^i_a = 0$ si, y solo si, $a \geq d_{reg,i}$, por lo que $*f_i$ es suprayectiva si, y solo si, $a \geq d_{reg,i}$. Por tanto, las condiciones de semi-regularidad se pueden expresar de la siguiente manera:

1. Primera definición: La sucesión $(f_1, \ldots, f_m)$ es semi-regular si para todo i se verifica que la aplicación lineal, multiplicar por f_i,

$$(S/I^{i-1})_{a-d_i} \xrightarrow{f_i} (S/I^{i-1})_a$$

es inyectiva para $0 \leq a < d_{reg,i}$. Ya que en otro caso la aplicación es suprayectiva por las consideraciones anteriores y, por tanto, de rango máximo.

2. Definición de Faugère: La sucesión $(f_1, \ldots, f_m)$ es semi-regular si para todo i se verifica que la aplicación lineal, multiplicar por f_i,

$$(S/I^{i-1})_{a-d_i} \xrightarrow{f_i} (S/I^{i-1})_a$$

es inyectiva para $0 \leq a < d_{reg}$.

Esta nueva visión de las definiciones da una relación entre ambas, a saber, la segunda es más débil ya que $d_{reg,i} \geq d_{reg}$. Veamos por casos cómo son las sucesiones semi-regulares

- Cuando $m < n$ se tiene que $d_{reg,i} = \infty$, ya que si no fuera así eso querría decir que con $i < n$ polinomios homogéneos se tiene que $V(f_1, \ldots, f_m) = \emptyset$, lo cual es imposible ya que como mínimo se necesitan n polinomios y, además, deberían formar una intersección completa (formar una sucesión regular). Por tanto, cuando $m < n$ ambas definiciones coinciden con la definición de regular o intersección completa.

- Si $m = n$ y la sucesión es semi-regular entonces se sabe que $(f_1, \ldots, f_{m-1})$ es regular. Por otro lado se tiene que $d_{reg,m} < \infty$, ya que sino la sucesión sería automáticamente regular y, además, $V(I) \neq \emptyset$, con lo que f_m se anularía en algunos puntos de $V(I^{m-1})$; lo cual está en contradicción con ser regular ya que existiría un g que se anulase en los demás puntos y, por tanto, la aplicación $*f_m$ no sería inyectiva. Suponiendo que $d_{reg,m} < \infty$ y que f_m es semi-regular en S/I^{m-1} se verifica que $*f_m$ es inyectiva para todo a (f_m regular en S/I^{m-1}); ya que si se supone lo contrario entonces existiría un $g \notin I^{m-1}$ tal que $f_m g \in I^{m-1}$, i.e, $f_m g$ se anula en los puntos de $V(I^{m-1})$, con lo que f_m se tendría que anular en algunos puntos, no todos, de $V(I^{m-1})$ lo cual implicaría que $d_{reg,m} = \infty$. Por tanto, en este caso una sucesión semi-regular también es regular. Supongamos ahora que la sucesión es semi-regular según la definición de Faugère, entonces análogamente al razonamiento anterior se tiene que $d_{reg} < \infty$. Geométricamente un elemento f_i es regular en S/I^{i-1} si, y solo si,

$$\dim(V(I^{i-1})) - 1 = \dim(V(I^{i-1}))$$

Como se tiene que $d_{reg} < \infty$ y $n = m$ formas, cada forma f_i necesariamente tiene que bajar 1 la dimensión y, por tanto, ser regular en S/I^{i-1}. Con lo que finalmente se llega a que en el caso $m = n$ ambas definiciones coinciden con la definición de

regular. Estos argumentos muestran que I está formado por una sucesión regular de polinomios si, y solo si, $V(I)$ es una intersección completa.

- Cuando $m > n$ las definiciones no son equivalentes, luego se verá cómo las series de Hilbert asociadas no tienen porqué ser iguales. Una observación inmediata es que

$$d_{reg,i} \geq d_{reg,i+1} \geq d_{reg}$$

ya que al añadir más formas nunca se puede disminuir la dimensión del espacio generado por dichas formas. Los argumentos geométricos anteriores muestran que cuando $m > n$ no puede haber sucesiones regulares, ya que llega un momento en el que no se puede seguir disminuyendo la dimensión. La definición de semi-regular y las consideraciones previas aseguran que los n primeros polinomios forman una sucesión regular.

Con estas consideraciones se puede pensar en sucesiones semi-regulares solo cuando $m > n$, y en dicho caso los ideales serán cero dimensionales con $V(I) = \{(0, \ldots, 0)\}$.

Se trabajará con la serie de Hilbert asociada a nuestro ideal, con la cual se pueden obtener relaciones entre las definiciones anteriores y se puede calcular de manera eficiente la complejidad del cálculo de las Bases de Groebner.

3.2. Propiedades

Veamos ahora ciertas propiedades que verifican la sucesiones regulares y semi-regulares. Gracias a esas propiedades vamos a poder entender mejor cómo se comporta la complejidad de algoritmos que dependen del d_{reg} como el F4. Para ello vamos a utilizar la Serie de Hilbert asociada al ideal generado por nuestras ecuaciones.

Definición 3.20. Se define la Serie de Hilbert del ideal I en A como:

$$HS_{A,I} = HS_I = \sum_{l=0}^{\infty} HF_{A,I}(l) z^l$$

Proposición 3.21. *Sea $I = (f_1, \ldots, f_m)$ un ideal homogéneo con $\deg f_i = d_i$, entonces $(f_1, \ldots, f_m)$ forman una sucesión regular si, y solo si, la serie de Hilbert viene dada por:*

$$HS_I = \sum_{k=1}^{\infty} c_k z^k = \frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1-z)^n}$$

Lema 3.22. *Sea A un anillo y $f \in A$. El elemento f es regular si, y solo si, $HS_{(f)} = (1-z^d)HS_0$.*

Demostración. El elemento f es regular si, y solo si,

$$0 \to A_{a-d} \xrightarrow{*f} A_a \to (A/f)_a \to 0$$

es una sucesión exacta para todo a. La condición anterior se da si, y solo si, para todo a se tiene que

$$\dim (A/f)_a = \dim A_a - \dim A_{a-d}$$

Cuando se quiere calcular la serie se tiene que ver cuánto aporta cada sumando de la expresión anterior a la serie. Se tienen las siguientes relaciones:

- $\dim (A/f)_a$ aporta $HS_{(f)}$.
- $\dim(A)_a$ aporta HS_0.
- $\dim(A)_{a-d}$ aporta $HS_0 t^d$, ya que el $-d$ hace que la serie HS_0 se vea desplazada d posiciones a la derecha y, por tanto, el primer coeficiente no nulo de la serie sea el d-ésimo.

Por tanto, f es regular si, y solo si, $HS_{(f)} = (1-z^d)HS_0$. □

Demostración de la proposición. Sea $I = (f_1, \ldots, f_m)$, $A = S/(f_1, \ldots, f_r)$ y $f = f_{r+1}$. Si se aplica inducción y el lema anterior se obtiene que I está formada por una sucesión regular si, y solo si,

$$HS_I = \prod_{i=1}^{m} (1-z^{d_i}) HS_0.$$

Se tiene que $HF_0(d) = \binom{n+d-1}{d}$, por lo que

$$HS_0 = \sum_{d=0}^{\infty} \binom{n+d-1}{d} z^d$$

Se puede comprobar que

$$HS_0 \cdot (1-z)^n = 1$$

por tanto se ha llegado a que

$$HS_I = \frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}.$$

□

Observación 3.23. Todos los argumentos valen si en vez de tomar el anillo S tomamos un subanillo A, aunque habría que comprobar que las formas siguen siendo regulares, ya que depende del anillo donde se trabaje.

Proposición 3.24.

1. *La propiedad de ser regular queda invariante por la permutación de elementos.*
2. *Si $(f_1, \ldots, f_m)$ es una sucesión regular entonces $(f_{i_1}, \ldots, f_{i_{m'}})$ es también una sucesión regular con $i_j \neq i_{j'}$ si $j \neq j'$ e $i_j \in \{1, \ldots, m\}$.*

Demostración.

1. Es inmediato al aplicar la proposición 3.21, ya que la serie de Hilbert no depende del orden de los polinomios.
2. Solo se tiene que aplicar el apartado uno y la definición para ver que $f_{i_{j+1}}$ es regular en $S/(f_{i_1}, \ldots, f_{i_j})$.

□

Ejemplo 3.25.

1. Supongamos que $I = (f_1, f_2)$ está formada por una sucesión regular. Si se aplica el lema 3.22, se obtiene que $HS_I = HS_0(1 - z^{d_1})(1 - z^{d_2})$. Si se quiere calcular a mano se tiene la siguiente sucesión exacta larga

$$\begin{array}{ccccccccccc} 0 & \to & S(-d_1-d_2) & \to & S(-d_1)\oplus S(-d_2) & \xrightarrow{j} & S & \xrightarrow{i} & S/I & \to & 0 \\ & & h & \to & (hf_2, hf_1) & & & & & & \\ & & & & (f,g) & \to & ff_1+gf_2 & & & & \end{array}$$

   Como los polinomios f_1 y f_2 forman una sucesión regular las únicas relaciones que hay entre ellos son las triviales. Se comprueba que la graduación es correcta ya que $(S(-d))_i = S_{i-d}$, es decir, si f tiene grado r en S entonces f tiene grado $r + d$ en $S(-d)$. Por tanto, se obtiene

$$\dim(S/I)_l = \dim S_l - \dim S_{l-d_1} - \dim S_{l-d_2} + \dim S_{l-d_1-d_2}$$

Al igual que antes hay que ver cuanto aporta cada sumando a la serie, en este caso se tiene que $\dim S_{l-d}$ aporta a la serie $HS_0 z^d$. Por tanto, se llega a:

$$HS_I = HS_0 - HS_0 z^{d_1} - HS_0 z^{d_2} + HS_0 z^{d_1+d_2} = HS_0(1-z^{d_1})(1-z^{d_2}).$$

2. Otra forma de calcular $HS_{(f)}$ cuando f es regular sería la siguiente. Dada la siguiente sucesión exacta

$$0 \to fA \to A \to A/fA \to 0$$

tomando grados se obtiene que:

$$\dim (A/fA)_l = \dim A_l - \dim (fA)_l$$

Ahora bien, como f es regular en A se tiene que el número de monomios de grado l en fA es el número de monomios de grado $l-d$ en A, por tanto

$$\dim (A/fA)_l = \dim A_l - \dim A_{l-d} \Rightarrow HS_I = HS_0(1-z^d)$$

3. Como ya se ha visto, la serie de Hilbert asociada a sucesiones regulares viene expresada de la siguiente manera:

$$HS_I = \frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}$$

Cuando $m < n$ se tiene que dicha serie no es un polinomio; esto ya se sabe porque el d_{reg} se puede ver como el primer índice de la serie cuyo coeficiente no es positivo, como $d_{reg} = \infty$ se tiene que la sucesión de los coeficientes no es eventualmente cero y, por tanto, no es un polinomio. En cambio cuando $m = n$ se tiene que dicha serie es un polinomio, de hecho se puede calcular gracias a los polinomios ciclotómicos.

Se define el k-ésimo polinomio ciclotómico como:

$$\Phi_k = \prod_{\zeta}(X-\zeta)$$

donde ζ recorre el conjunto de las raíces k-ésimas primitivas de la unidad. Se sabe que el polinomio $X^n - 1$ tiene como solución todas las raíces n-ésimas de la unidad, por tanto tendrá como solución todas las raíces k-ésimas primitivas de la unidad con

k/n. De hecho esas son todas sus soluciones ya que

$$n = \sum_{k/n} \varphi(k)$$

Una de las propiedades básicas de los polinomios ciclotómicos es que pertenecen a $\mathbb{Z}[X]$, son mónicos e irreducibles. Si p es un número primo, por lo visto anteriormente, se tiene que

$$X^p - 1 = \prod_{k/p} \Phi_k = \Phi_1 \Phi_p$$

por lo que se obtiene

$$\Phi_p = \frac{X^p - 1}{X - 1} = X^{p-1} + X^{p-2} + \cdots + X + 1$$

Con estas relaciones se llega a lo siguiente

$$HS_I = \frac{\prod_{i=1}^{n}(1 - z^{d_i})}{(1-z)^n} = \prod_{i=1}^{n} \frac{\prod_{k/d_i} \phi_k}{1 - z} = \prod_{i=1}^{n} \prod_{\substack{k/d_i \\ k \neq 1}} \Phi_k$$

Dicho polinomio tiene grado $\sum_{i=1}^{n}(d_i - 1)$ por tanto $d_{reg} = \sum_{i=1}^{n}(d_i - 1) + 1$.

Observación 3.26. Otra observación importante que se puede deducir de lo anterior es que si tienen dos ideales I e I' donde ambos ideales están formados por sucesiones de polinomios con el mismo número de polinomios, el mismo grado y el mismo número de variables entonces, si la primera sucesión de polinomios asociada al ideal I es regular, se obtiene que:

$$HF_I(l) \leq HF_{I'}(l).$$

Lo cual implica que las sucesiones regulares son aquellas cuyos polinomios tienen el menor número de relaciones internas. En particular esto asegura que

$$d_{reg}(I) \leq d_{reg}(I').$$

Veamos las propiedades que tienen las series de Hilbert cuando nuestro ideal está formado por una sucesión semi-regular de polinomios homogéneos.

Definición 3.27. Sea $S(z)$ una serie. Definimos la serie truncada $[S(z)] = \sum_{i=0}^{\infty} b_i z^i$ como:

$$b_i = \begin{cases} a_i & a_j > 0 \quad \forall j \leq i \\ 0 & \text{en otro caso} \end{cases}$$

Teorema 3.28. *Sea $I = (f_1, \ldots, f_m)$ un ideal homogéneo con $\deg f_i = d_i$, entonces $(f_1, \ldots, f_m)$ forman una sucesión semi-regular si, y solo si, la serie de Hilbert viene dada por:*

$$HS_{(f_1,\ldots,f_s)} = \sum_{k=1}^{\infty} c_k z^k = \left[\frac{\prod_{i=1}^{s}(1 - z^{d_i})}{(1-z)^n} \right]$$

para todo $1 \leq s \leq m$.

Demostración. Sea $I^i = (f_1, \ldots, f_i)$ y $A^i = S/I^i$. Para cada i se tiene la siguiente sucesión exacta larga:

$$\ker(*f_i) \to A^{i-1}_{a-d_i} \overset{*f_i}{\to} A^{i-1}_a \to A^i_a \to 0$$

Esto quiere decir que $\dim A^i_a = \dim A^{i-1}_a - \dim A^{i-1}_{a-d_i} + \dim(\ker(*f_i))$. Se tienen por tanto 3 posibilidades:

1. Si $*f_i$ es inyectiva, entonces

$$\dim A^i_a = \dim A^{i-1}_a - \dim A^{i-1}_{a-d_i}$$

2. Si $*f_i$ es suprayectiva, ya se ha visto antes que

$$\dim A^i_a = 0$$

3. Si $*f_i$ no tiene rango máximo, entonces

$$\dim A^i_a > \text{máx}\left\{ \dim A^{i-1}_a - \dim A^{i-1}_{a-d_i}, 0 \right\}$$

Supongamos que la sucesión es semi-regular; se tiene que f_i es semi-regular en A^{i-1}. Lo que quiere decir que los primeros coeficientes de la serie $\dim A^{i-1}_a - \dim A^{i-1}_{a-d_i}$ vienen dados por los coeficientes de la serie $(1 - z^{d_i})HS_{I^{i-1}}$. Hasta un cierto índice, $d_{reg,i}$, a partir del cual los coeficientes serán 0 ya que el coeficiente no positivo $\dim A^{i-1}_a - \dim A^{i-1}_{a-d_i}$ indica

$-\dim(\ker(*f_i))$. Por tanto,

$$\sum_{a=0}^{\infty} \text{máx}\{\dim A_a^{i-1} - \dim A_{a-d_i}^{i-1}, 0\} z^a = \left[(1-z^{d_i}) HS_{I^{i-1}}\right]$$

Observación 3.29. Si se conoce la serie $HS_{I^{i-1}}$ y la multiplicamos por $(1-z^{d_i})$, los coeficientes negativos están indicando $-\dim(\ker(*f_i))$, si el elemento f_i es regular en A^{i-1}.

Sea i el mínimo valor para el cual la función $*f_i$ no tiene rango máximo. En este caso no se puede dar la igualdad anterior, ya que la serie tendrá al menos un coeficiente distinto, por tanto para que se cumpla la expresión $\left[(1-z^{d_i}) HS_{I^{i-1}}\right]$ es necesario que f_i sea regular en A^{i-1}. Lo único que falta es ver que se cumple la siguiente condición:

$$\left[(1-z^d) S(z)\right] = \left[(1-z^d)\,[S(z)]\right]$$

donde $S(z)$ es una serie. Sean $(a_0, a_1, \dots)$ los coeficientes asociados a la serie $S(z)$ y $(a_0, a_1, \dots, a_n, 0, \dots)$ los coeficientes asociados a la serie truncada $[S(z)]$. En particular esto indica que a_{n+1} es no positivo. Los coeficientes de $(1-z^d)S(z)$ son:

$$(a_0, a_1, \dots, a_d - a_0, \dots a_n - a_{n-d}, a_{n+1} - a_1, \dots)$$

mientras que los coeficientes de la serie $(1-z^d)\,[S(z)]$ son

$$(a_0, a_1, \dots, a_d - a_0, \dots a_n - a_{n-d}, -a_1, \dots)$$

Ambas sucesiones coinciden en los n primeros términos y, además, en los dos casos el termino $n+1$ es no positivo, de manera que al truncar ambas series se obtiene el mismo resultado. Por tanto, con las consideraciones anteriores y aplicando inducción, se obtiene el resultado. □

Observación 3.30. Esta proposición asegura que si $(f_1, \dots, f_m)$ es semi-regular entonces $(f_1, \dots, f_s)$ es semi-regular para todo $s < m$. Antes ya se ha visto que en este caso $(f_1, \dots, f_n)$ formaban una sucesión regular. Aunque en este caso hay contraejemplos en los que no se cumple que la propiedad de semi-regular quede invariante por permutaciones. Por ejemplo la sucesión (x^2, y^2, xy) es semi-regular mientras que (x^2, xy, y^2) no lo es. Con esta caracterización de las sucesiones semi-regulares es inmediato que que si $m \leq n$ entonces toda sucesión regular es semi-regular.

Proposición 3.31. *Sea $I = (f_1, \dots, f_m)$ un ideal homogéneo con $\deg f_i = d_i$, entonces*

$(f_1, \ldots, f_m)$ forman una sucesión semi-regular según la definición de Faugère si, y solo si, la serie de Hilbert viene dada por:

$$HS_I = \sum_{k=1}^{\infty} c_k z^k = \left[\frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1-z)^n} \right] \tag{3.2.1}$$

Antes de proceder con la demostración se necesita una definición y varios lemas.

Definición 3.32. Una sucesión de polinomios $(f_1, \ldots, f_m)$ es regular hasta el grado D si la aplicación lineal $*f_i$ es inyectiva para todo $1 \leq i \leq m$ y todo grado $0 \leq d \leq D$.

Lema 3.33. *Sea $[S]_d$ la serie obtenida a partir de truncar la serie S por el grado $d+1$. Se verifica entonces que, para cualesquiera dos series S y T, se tiene*

$$[[S]_d [T]_d]_d = [S \cdot T]_d$$

Demostración. Para conocer los coeficientes de $S \cdot T$ hasta el índice d solo es necesario conocer los coeficientes de S y T hasta el índice d. □

Lema 3.34. *Si f es regular en A hasta el grado D entonces se verifica que:*

$$HS_{(f)} \equiv (1 - t^d) HS_0 \mod t^{D+1}$$

Demostración. Este lema se demuestra utilizando el lema 3.22 y el lema anterior. □

Demostración de la proposición 3.31. Procedamos primero con la demostración de izquierda a derecha. Aplicando el lema anterior e inducción se llega a que

$$HS_I = \frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1-z)^n} \mod t^{d_{reg}}.$$

Solo faltaría ver que el coeficiente d_{reg}-ésimo de la serie

$$S(n, m) = \frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1-z)^n}$$

es no positivo (la ecuación anterior dice que todos los coeficientes anteriores son positivos). Sea $c_k^{n,m}$ el coeficiente k-ésimo de la serie $S(n, m)$. Se tiene que

$$A_{d_{reg}-d_m}^{m-1} \xrightarrow{*f_m} A_{d_{reg}}^{m-1}$$

a partir de la definición de d_{reg} se llega a que

$$\dim A^{m-1}_{d_{reg}} - \dim A^{m-1}_{d_{reg}-d_m} \leq \dim A^m_{d_{reg}} = 0$$

Se tiene que $\dim A^{m-1}_{d_{reg}-d_m}$ es $c^{n,m-1}_{d_{reg}-d_m}$, ya que $d_m > 0$, y

$$\dim A^{m-1}_{d_{reg}} \leq \dim A^{m-2}_{d_{reg}} - \dim A^{m-2}_{d_{reg}-d_{m-1}}$$

Aplicando este proceso reiteradamente se llegará a un momento en el que

$$\dim A^{m-i}_{d_{reg}} = \dim A^{m-i-1}_{d_{reg}} - \dim A^{m-i-1}_{d_{reg}-d_{m-i}} = c^{n,m-i}_{d_{reg}}$$

ese momento será cuando $d_{reg,m-i} \geq d_{reg}$. Llamamos α^m_d a $\dim A^m_d$. Visualmente se obtiene

$$\begin{array}{ccccc}
\alpha^m_{d_{reg}} & & & & \\
\vee| & & & & \\
\alpha^{m-1}_{d_{reg}} & - & \alpha^{m-1}_{d_{reg}-d_m} & (= c^{n,m-1}_{d_{reg}-d_m}) & \\
\vee| & & & & \\
\alpha^{m-2}_{d_{reg}} & - & \alpha^{m-2}_{d_{reg}-d_{m-1}} & (= c^{n,m-2}_{d_{reg}-d_{m-1}}) & \\
\vee| & & & & \\
\vdots & & \vdots & & \\
\vee| & & & & \\
\alpha^{m-i}_{d_{reg}} & - & \alpha^{m-i}_{d_{reg}-d_{m-i+1}} & (= c^{n,m-i}_{d_{reg}-d_{m-i+1}}) & \\
|| & & & & \\
c^{n,m-i}_{d_{reg}} & & & &
\end{array}$$

se ha obtenido que

$$0 = \alpha^m_{d_{reg}} \geq \left(\left(\left(\left(c^{n,m-i}_{d_{reg}} - c^{n,m-i}_{d_{reg}-d_{m-i+1}}\right) - c^{n,m-i+1}_{d_{reg}-d_{m-i+2}}\right)\ldots\right) - c^{n,m-1}_{d_{reg}-d_m}\right) = c^{n,m}_{d_{reg}}.$$

Para la demostración de derecha a izquierda solo hace falta ver que

$$HS_I \equiv \frac{\prod_{i=1}^m (1-z^{d_i})}{(1-z)^n} \quad \text{mód } t^{d_{reg}}.$$

Supongamos que se tienen dos series S y T tal que $S >^l T$ (orden lexicográfico), entonces fácilmente se ve que

$$(1-t^d)S >^l (1-t^d)T. \tag{3.2.2}$$

Además, el primer índice en el que ambas series son distintas es igual en ambos casos. Con esta consideración supongamos que $(f_1, \ldots, f_m)$ no es regular hasta el grado d_{reg}. Se toma el menor q de manera que $(f_1, \ldots, f_q)$ no es regular hasta grado d_{reg}. En este caso se comprueba que

$$HS_{(f_1,\ldots,f_q)} >^l S(n,q) \quad \text{mód } t^{d_{reg}}. \tag{3.2.3}$$

Ya que justamente el primer coeficiente en el que las series difieren es aquel en el que la aplicación $*f_q$ no es inyectiva y, por tanto, la diferencia entre ambos coeficientes será $\dim(\ker(*f_q))$. Además, siempre se tiene que

$$HS_{(f_1,\ldots,f_r)} \geq^l HS_{(f_1,\ldots,f_{r-1})}(1-t^{d_r}) \quad \text{mód } t^{d_{reg}}. \tag{3.2.4}$$

Por tanto, utilizando las últimas 3 desigualdades se llega a que si $(f_1, \ldots, f_m)$ no es regular hasta grado d_{reg} entones necesariamente se tiene que

$$HS_{(f_1,\ldots,f_m)} <^l S(n,m) \quad \text{mód } t^{d_{reg}}.$$

□

En particular este teorema dice que

$$HS_{(f_1,\ldots,f_m)} \geq^l [S(n,m)]$$

Observación 3.35. Esta proposición asegura que la propiedad de ser semi-regular, según la definición de Faugère, queda invariante por permutación. Aunque con la definición de Faugère no es cierto que: si $(f_1, \ldots, f_n)$ es semi-regular entonces $(f_1, \ldots, f_s)$ sea semi-regular para todo $s < n$. Por ejemplo, la sucesión (x^2, xy, y^2) es semi-regular con la definición de Faugère mientras que (x^2, xy) no es semi-regular con dicha definición. Ahora es sencillo ver que la definición de semi-regularidad dada por Faugère es más débil que la definición de semi-regularidad.

3.3. Conclusiones

Veamos ahora qué conclusiones podemos obtener utilizando todas las propiedades mostradas anteriormente.

Llegados a este punto ya se tiene una manera eficiente de calcular el índice de regularidad para sucesiones semi-regulares. Para calcular el índice de regularidad de una sucesión

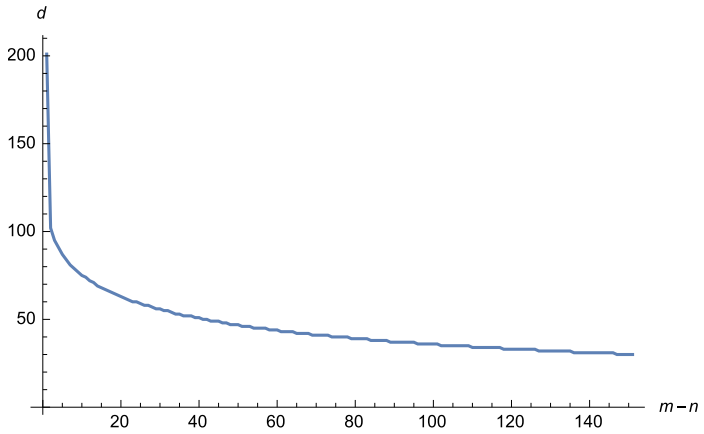

Figura 3.3.1: d_{reg} en función de $m - n$, para sistemas de 100 variables cuadráticos.

$(f_1, \ldots, f_m)$ solo se tiene que calcular el índice k del primer coeficiente c_k no positivo de la serie

$$\frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1 - z)^n}$$

Esto se puede computar de una manera no muy difícil. En particular el índice de regularidad depende solo del número de variables, el número de ecuaciones y los grados de cada uno de los polinomios involucrados, siempre y cuando la sucesión sea semi-regular. Simplemente con la definición de d_{reg} se ve que el índice de regularidad disminuye a medida que vamos añadiendo polinomios semi-regulares a la sucesión. En la figura 3.3.1 se ve como a medida que añadimos ecuaciones el d_{reg} va disminuyendo.

Que una sucesión de polinomios sea semi-regular en particular dice que los polinomios son linealmente independientes, de hecho se verifica que no hay ningún tipo de relación con coeficientes polinomiales entre ellos hasta grado d_{reg}. A partir de dicho grado las relaciones son inevitables, ya que dichos polinomios verifican que

$$(f_1, \ldots, f_m)_k = S_k$$

para $k \geq d_{reg}$, por lo que llega un momento en el que hay relaciones entre ellos que no se pueden evitar. Además, se tiene que no existe otra configuración de polinomios con las mismas características que puedan salvar dichas relaciones a partir del grado d_{reg} ya que, al igual que en el caso de sucesiones regulares, se tiene que con cualquier otra sucesión $I' = (f'_1, \ldots, f'_m)$ se verifica que:

$$HF_I(l) \leq HF_{I'}(l)$$

como se verá inmediatamente.

Por tanto, decir que un sistema de polinomios forma una sucesión semi-regular es lo mismo que decir que dichos polinomios son lo más independientes posible con ese orden.

Supongamos que se tiene un ideal I formado por una sucesión semi-regular de polinomios, si intercambiamos uno de esos polinomios por otro de manera que la sucesión no sea semi-regular, esto afectará directamente a la serie de Hilbert del ideal. Al hacer el cambio lo que se produce es que a cada coeficiente c_k se le suma $\dim\left(\ker\left(*f_i^k\right)\right)$. Como consecuencia la serie resultante es mayor o igual que la serie que teníamos y el nuevo índice de regularidad d'_{reg} verifica que $d'_{reg} \geq d_{reg}$. Por tanto, se tiene el siguiente teorema.

Teorema 3.36. *Sea $I = (f_1, \ldots, f_m)$ un ideal generado con $\deg f_i = d_i$. Se verifica que:*

$$HS_I \geq \left[\frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}\right]$$

donde n es el número de variables.

Demostración. [Die15] □

Hay ciertos programas que calculan el índice de regularidad de un ideal, pero hay que tener cuidado. Por ejemplo, el índice de regularidad que Sage calcula se obtiene a partir de la fórmula anterior suponiendo que la sucesión sea semi-regular, eso quiere decir que si no introducimos una sucesión semi-regular el valor que devuelve Sage no se corresponde con el índice de regularidad. No hay una fórmula explícita para el calculo del índice de regularidad en función de los parámetros descritos anteriormente, lo que sí se puede hacer es calcular su valor en cada caso concreto.

Una vez conocido el valor d_{reg} para un ideal se tiene que la complejidad del F4 para una sucesión semi-regular viene dada por:

$$O\left(\binom{n+d_{reg}}{n}^w\right) \tag{3.3.1}$$

ya que en cada etapa d solo se trabaja con polinomios homogéneos de grado d, por tanto para $d > d_{reg}$ el algoritmo no modifica la base G y G será una base de Groebner para el ideal I. Con esta fórmula y el teorema 3.36 nos damos cuenta de que cuanto más independientes sean nuestros polinomios o más polinomios añadamos, la complejidad decrece porque los coeficientes de la serie de Hilbert disminuyen. Es decir, si se tiene $I = (f_1, \ldots, f_m)$ y

$I' = (f_1, \ldots, f_m, f_{m+1})$ entonces se comprueba fácilmente que

$$HF_{I'}(l) \leq HF_I(l).$$

Y en particular, como en casos anteriores, se tiene que $d_{reg}(I') \leq d_{reg}(I)$, lo cual se puede ver experimentalmente.

La ecuación 3.3.1 hace una buena estimación de la complejidad del F4. En la parte izquierda de la imagen 3.3.2 se muestra la complejidad del algoritmo F4 para sistemas homogéneos y cuadráticos donde el número de variables n varía entre 3 y 14, y el número de ecuaciones m varía entre $n+1$ y 28 en $\mathbb{F}_3$ (los tiempos están truncados).

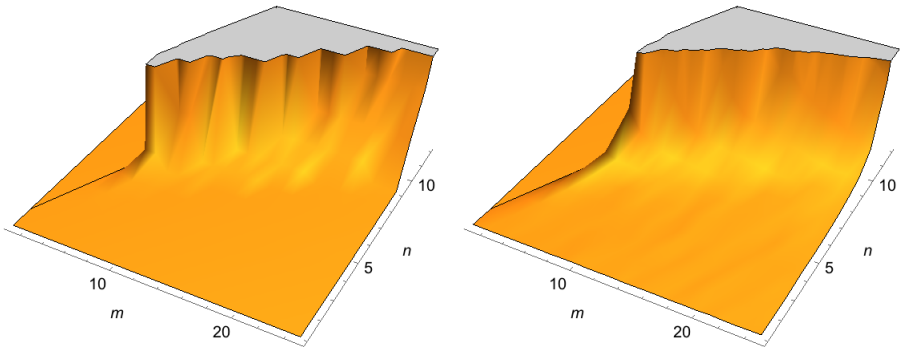

Figura 3.3.2: Complejidad teórica vs tiempo real de ejecución.

En cambio, en la parte derecha se muestra el tiempo real de ejecución para sistemas aleatorios con los mismos parámetros que antes. Las discrepancias entre ambas figuras se deben a que, en la práctica, sistemas con el mismo número de variables y el mismo índice de regularidad tienen distinta complejidad. Por ejemplo puede ser que al añadir una ecuación (y preservar la propiedad de ser un sistema semi-regular) a nuestro sistema el índice de regularidad no varíe, en cambio al añadir la ecuación conseguimos que la HS_I sea menor o igual que la que teníamos y con menos operaciones conseguimos un conjunto G de manera que $LT(G)$ genere $S_{d_{reg}}$. Este hecho se aprecia bastante bien en la figura 3.3.3, donde se ha representado el tiempo de ejecución y el d_{reg} para sistemas homogéneos, cuadráticos y aleatorios en $\mathbb{F}_3$ con 11 variables. Se puede apreciar que, aunque el índice de regularidad se mantenga, la complejidad disminuye. Además, también se puede observar el carácter exponencial del algoritmo F4. Toda esta teoría es una aproximación para obtener información del funcionamiento de las bases de Groebner para sistemas arbitrarios. ¿En qué sentido es una aproximación? Lo primero es que nos estamos restringiendo al uso de polinomios

homogéneos, lo cual en principio no debe ser un problema para el estudio de la complejidad de las bases de Groebner. Que sean ideales cero dimensionales es natural pedirlo ya que trabajamos con sistemas sobredefinidos y lo normal es que dichos sistemas tengan solo una solución (o muy pocas).

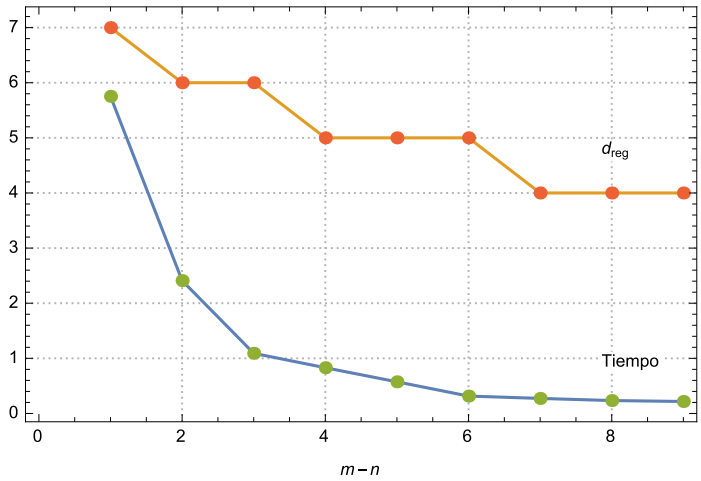

Figura 3.3.3: Complejidad y d_{reg} en función de $m - n$.

Considerar estas dos restricciones no parece que vaya a cambiar mucho la complejidad, es decir, dichos sistemas tienen la misma complejidad que sistemas aleatorios, pero ¿Qué pasa con el hecho de pedir que nuestro sistema forme una sucesión semi-regular? Se sabe que en el caso de característica 0 y fijado el número de variables y ecuaciones el ser regular es genérico. Esto se puede traducir diciendo que casi todo sistema es regular o que el ser regular caracteriza bien la noción de sistema aleatorio o que la probabilidad de elegir aleatoriamente un sistema no regular es 0. En este punto es cuando se hará la primera aproximación, al suponer que, en el caso de sistemas sobredefinidos, se sigue verificando esta propiedad, lo cual es una conjetura que está relacionada con la conjetura de Fröberg. La conjetura de Fröberg dice que toda sucesión genérica de polinomios tiene una serie de Hilbert igual a la ecuación 3.2.1. Lo que se hará, por tanto, es suponer que para cuerpos finitos y sistemas sobredefinidos se tiene que los sistemas semi-regulares son genéricos o que caracterizan a los sistemas aleatorios. Que se puede traducir diciendo que a medida que el número de variables y ecuaciones aumenta se verifica que la probabilidad de que un sistema sea semi-regular tiende a 1. Como se está trabajando en un cuerpo finito el decir que un conjunto es genérico no nos dice mucho, ya que todo conjunto es genérico. Pero se espera que dicha propiedad, como en el caso regular de característica 0, caracterice bien la

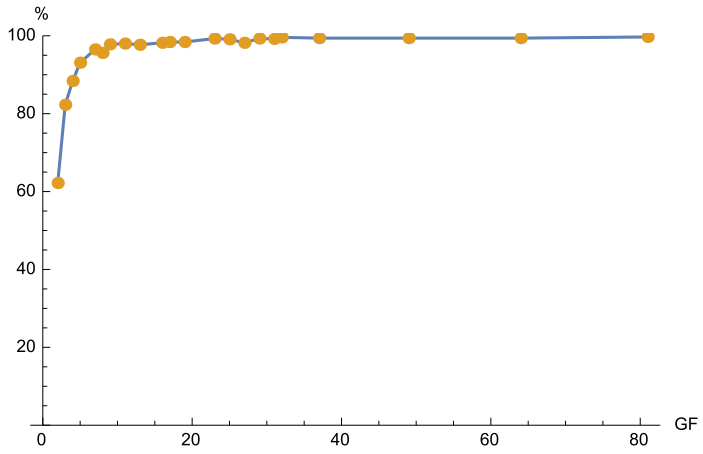

Figura 3.3.4: Porcentaje de sucesiones semi-regulares en función del tamaño del cuerpo.

noción de sistema aleatorio.

En el caso de cuerpos infinitos se puede establecer la conjetura de la siguiente manera.

Conjetura 3.37. *Sea* $\{f_1, \ldots, f_m \mid deg(f_i) = d_i\} = S_{d_1} \times \cdots \times S_{d_m}$ *y* K *un cuerpo infinito. El conjunto de sucesiones semi-regulares es un abierto no vacío en la topología de Zariski del espacio* $S_{d_1} \times \cdots \times S_{d_m}$.

Experimentalmente sobre cuerpos finitos se ve que la mayoría de los sistemas homogéneos de ecuaciones están formados por sucesiones semi-regulares. Como se puede apreciar en la figura 3.3.4, el porcentaje de sucesiones semi-regulares tiende a 100 % a medida que aumentamos el tamaño del cuerpo. La gráfica ha sido generada con muestras aleatorias.

De la misma manera, a medida que aumenta el número de variables aumenta el porcentaje aun dejando el cuerpo fijo. Como se ve en la figura 3.3.5, si se consideran sistemas de ecuaciones cuadráticos sobre $\mathbb{F}_2$ a medida que el número de variables aumenta el porcentaje crece.

Observación 3.38. Para casos sencillos el índice de regularidad se puede calcular de manera más simple.

- $d_i = 2$ y $m = n$. En este caso la Serie de Hilbert viene dada por

$$HS = \frac{\left(1 - z^2\right)^n}{(1 - z)^n} = \frac{(1 - z)^n (1 + z)^n}{(1 - z)^n} = (1 + z)^n$$

  Por lo que el índice de regularidad es $n + 1$.

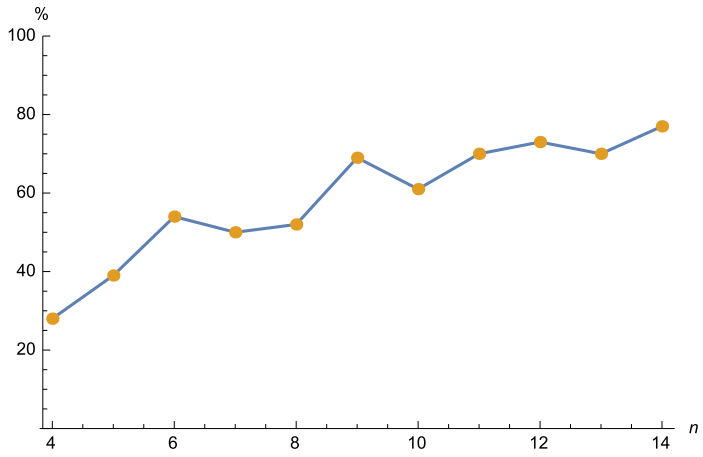

Figura 3.3.5: Porcentaje de sucesiones semi-regulares en función del número de variables.

- $d_i = 2$ y $m = n + 1$. En este caso se tiene

$$HS = (1 - z)(1 + z)^m$$

Cuyos coeficientes son de la forma $\binom{m}{i} - \binom{m}{i-1}$. Si dibujamos el triángulo de Tartaglia nos damos cuenta de que el coeficiente es no positivo justamente cuando el i es mayor estricto que $m + 1$. Por tanto, si m es par se tiene que $d_{reg} = \frac{m}{2} + 1$ y si m es impar $d_{reg} = \frac{m+1}{2}$. En este caso, tomando la complejidad dada por Faugère, se tiene que el problema se podrá resolver en

$$\begin{aligned}
O\left(\binom{n + d_{reg}}{n}^w\right) &= O\left(\binom{n + n/2 + 1}{n}^w\right) = \\
&= O\left(\left(\frac{1}{\sqrt{2\pi n}}\left(1 + \frac{n}{n/2+1}\right)^{\frac{n/2+2}{2}}\left(\frac{3n/2+1}{n}\right)^n\right)^w\right) \\
&= O\left(\left(n^{-\frac{1}{2}} 3^{\frac{n}{4}}\left(\frac{3}{2}\right)^n\right)^w\right) \\
&= O\left(2^{\left(-\frac{1}{2}\log n + \frac{n}{4}\log 3 + n\log\frac{3}{2}\right)w}\right) \\
&\approx O\left(2^{n\left(\frac{\log 3}{4} + \log\frac{3}{2}\right)w}\right) = O\left(2^{2{,}65\cdot n}\right)
\end{aligned}$$

Esta es una complejidad que se acerca bastante bien a los experimentos de Faugère.

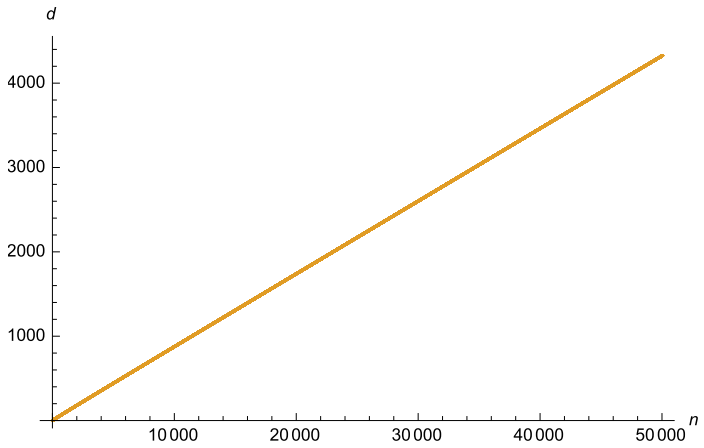

Figura 3.3.6: d_{reg} para sistemas cuadráticos con n variables y $2n$ ecuaciones y aproximación dada por Faugère.

- $d_i = 2$ y $m = 2n$. En este caso se tiene

$$HS = (1-z)^n (1+z)^{2n}$$

por lo que el coeficiente a_k de la Serie de Hilbert será

$$\sum_{i=\text{máx}\{0,k-2n\}}^{\text{mín}\{k,n\}} (-1)^i \binom{n}{i} \binom{2n}{k-i}$$

o 0 si la suma resulta ser negativa. Por tanto, el índice de regularidad será el menor k para el cual dicha suma sea no positiva. En este caso Faugère estima que el índice de regularidad se puede aproximar por

$$d_{reg} = 0{,}0858n + 1{,}04n^{1/3} - 1{,}47 + \frac{1{,}71}{n^{1/3}} + O\left(\frac{1}{n^{2/3}}\right)$$

Esta aproximación es bastante buena, de hecho, en la figura 3.3.6 se ve como la fórmula dada por Faugère se solapa con el d_{reg} real.

- $d_i = 2$ en el caso general se tiene que a_k es

$$\sum_{i=\text{máx}\{0,k-m\}}^{\text{mín}\{k,m-n\}} (-1)^i \binom{m-n}{i} \binom{m}{k-i}$$

Con esta fórmula se puede calcular de manera muy rápida el d_{reg} de sistemas cua-

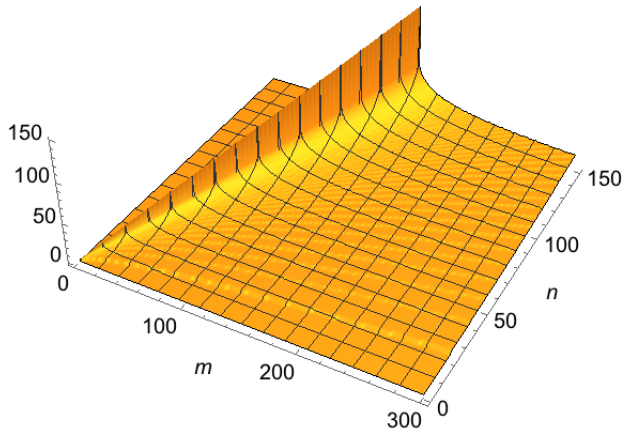

Figura 3.3.7: d_{reg} en función de n y m para sistemas cuadráticos.

dráticos. En la figura 3.3.7 se puede ver el d_{reg} en función de n y m para sistemas cuadráticos.

- Si $m = n$ entonces se puede ver, gracias a los polinomios ciclotómicos, que el $d_{reg} = n(d-1)+1$.

En los estudios de la complejidad siempre se supone que $d_{reg} < q$, por lo que las ecuaciones del cuerpo no se utilizan para optimizar el tiempo de ejecución de los algoritmos.

En principio para $d_i = 2$ y $m = 2n$ se puede apreciar experimentalmente que $d_{reg} = O(n)$.

Observación 3.39. Otros autores [CG17] llaman índice de regularidad al grado más alto de los polinomios que aparecen en una base de Groebner. En general esta definición no es equivalente a las anteriores. Con esta definición se puede aproximar bastante bien el índice de regularidad, ya que el índice de regularidad se puede ver que es una cota superior del grado máximo de los polinomios que aparecen en una base de Groebner. Además, experimentalmente se ve que para sistemas aleatorios generalmente el grado máximo en una base de Groebner del ideal y el índice de regularidad del ideal son el mismo. Por otra parte hay otro concepto importante y que también se utiliza, el cual es el siguiente:

Definición 3.40. Sea $I = \{f_1, \ldots, f_m\}$ un conjunto de polinomios sobre el cuerpo $\mathbb{F}$. Supongamos, sin pérdida de generalidad, que son homogéneos; en caso de no serlo trabajaríamos con la parte de mayor grado de los polinomios. Definimos el first fall degree de I como el

mínimo D_{ff} para el cual existe una relación no trivial de la forma

$$g_1 f_1 + \cdots + g_m f_m = 0$$

Como su nombre indica, el first fall degree es el primer grado en el que aparece una relación polinomial no trivial entre los polinomios de nuestro sistema. Cuando trabajamos con polinomios no homogéneos lo que aparece es una caída del grado máximo, es decir, la parte homogénea de mayor grado se cancela. Ahora veamos que relaciones hay entre el first fall degree y el índice de regularidad. Supongamos que se tiene un sistema de m ecuaciones y n incógnitas sobre el cuerpo $\mathbb{F}$, entonces se tiene lo siguiente:

- Si $m < n$ y la sucesión es regular entonces no existe D_{ff} ni d_{reg}.
- Si $m = n$ y la sucesión es regular entonces no existe D_{ff} y si existe d_{reg}.
- Si $m > n$ y la sucesión es semi-regular entonces $D_{ff} = d_{reg}$.
- Si la sucesión no es regular ni semi-regular, entonces se tiene que D_{ff} existe, además si d_{reg} existe también se tiene que $D_{ff} < d_{reg}$.

Estas relaciones se deducen de la Serie de Hilbert, del ideal asociado y de la definición de regular y semi-regular. Por otra parte los experimentos han mostrado a lo largo de los últimos años que si el algoritmo F4 se aplica a sistemas donde hay ciertas relaciones polinomiales, entonces cuando el algoritmo encuentra una caída de grado, o equivalentemente cuando llega al grado D_{ff}, el algoritmo rápidamente termina con lo que D_{ff} es muy similar o igual al grado máximo de los polinomios en la base encontrada. Por tanto, muchas veces se toman estos valores como medidas de la complejidad para el cálculo de Bases de Groebner.

Pero llegados a este punto tenemos que preguntarnos por qué se tienen hasta tres definiciones similares referentes a la complejidad de las bases de Groebner. La cuestión es la siguiente, el índice de regularidad es un dato teórico que solo da información sobre la complejidad de la base de Groebner cuando la sucesión es semi-regular (sucesión genérica). Pero justo en ese caso es cuando la complejidad se dispara por lo que tampoco aporta mucha información. En cambio cuando la sucesión no es semi-regular, es decir, cuando hay relaciones internas, el algoritmo termina cuando llega al grado D_{ff}. Pero en este caso calcular D_{ff} y una Base de Groebner tienen casi la misma complejidad por lo que no aporta información. Con esto quiero decir que en la práctica no se tiene una medida a

priori de la complejidad de un sistema de ecuaciones ya que hay dos extremos; o el sistema es semi-regular (sin relaciones internas) y es muy difícil calcular la base; o el sistema tiene relaciones internas y es fácil obtener una base. En ambos casos no se puede saber si tiene o no relaciones, porque para ello se tiene que calcular una base. Por tanto, la única manera de saber la complejidad es poner en marcha un algoritmo y esperar, pero es paradójico porque en ningún momento se sabe si al algoritmo le queda poco. Por tanto, de un sistema polinomial solo se puede decir que es fácil calcular una Base de Groebner, ya que en el caso de que sea difícil solo se sabrá habiendo calculado la base, lo cual no es posible (aunque se sabe que, en general, el cálculo de una Base de Groebner es difícil).

Observaciones finales

Si nos ayudamos de las ecuaciones del cuerpo $x_i^q - x_i$ para calcular la solución de un sistema de ecuaciones (introduciéndolas en el ideal), estas solo ayudaran cuando el cuerpo sea pequeño en relación con el índice de regularidad, ya que en ese caso podremos ayudarnos de dichas ecuaciones para hacer simplificaciones.

Para poder resolver sistemas de ecuaciones polinomiales utilizando las bases de Groebner se debe utilizar un orden lexicográfico, en cambio el F4 no utiliza ese orden. Pero existen algoritmos que pueden cambiar el orden de la base en tiempo polinomial, es decir, dada una base de Groebner respecto de un orden calculan una base de Groebner para otro orden dado [Bar09].

Capítulo 4

Criptografía Multivariable

Los sistemas criptográficos que están englobados dentro de la criptografía multivariable [DGS06, BBD09, Tho13, Wol05] son aquellos sistemas de clave pública cuya aplicación de cifrado o clave pública es un sistema de polinomios no lineales en varias variables. La seguridad de estos sistemas recae en la complejidad de resolver sistemas de ecuaciones polinomiales no lineales. El problema de resolver tales sistemas se sabe que es NP-duro. Aunque ello no garantiza la seguridad de dichos sistemas. En este capítulo mostraremos ciertas propiedades básicas y un ejemplo fundamental, el HFE [Pat96c].

4.1. Aplicaciones polinomiales

En esta sección se verán ciertas propiedades básicas que tienen las aplicaciones polinomiales. Como siempre el cuerpo base es un cuerpo finito de q elementos $\mathbb{F} = \mathbb{F}_q$.

Definición 4.1. El conjunto de los sistemas de polinomios cuadráticos en n variables con m ecuaciones sobre el cuerpo $\mathbb{F}$ se denotará por $\mathcal{MQ}(n, m)$.

Como ya se ha comentado anteriormente existe un isomorfismo

$$\phi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_{q^n},$$

por lo que en cualquier momento se puede pensar en un elemento de $\mathbb{K} = \mathbb{F}_{q^n}$ o en sus coordenadas de manera indistinta. Normalmente se representará con letras mayúsculas a los elementos de $\mathbb{K}$ y con letras minúsculas a los elementos de $\mathbb{F}$, i.e,

$$X = \phi(x) = \phi((x_1, \ldots, x_n))$$

Se empezará por las aplicaciones más sencillas, las afines.

Sea $F(X)$ una aplicación afín de $\mathbb{K}$ en $\mathbb{K}$. Como $\mathbb{K}$ es un $\mathbb{F}$-espacio vectorial de dimensión n se puede pensar en la representación matricial de $F(X)$, la cual tendrá la siguiente forma en coordenadas

$$F(x) = Ax + b,$$

con $A \in \mathcal{M}_{n\times n}(\mathbb{F})$ y $b \in \mathbb{F}^n$. También se puede ver de la siguiente manera

$$F(X) = \phi\left(A\left(\phi^{-1}(X)\right) + b\right).$$

Otra forma equivalente sería la representación coordenada a coordenada o multivariable

$$F(x) = \begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n + b_1 \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n + b_n \end{pmatrix}.$$

Gracias a la teoría de cuerpos finitos se sabe que los n Homomorfismos de Frobenius forman una base del espacio de las aplicaciones lineales sobre $\mathbb{K}$. Esto quiere decir que cualquier aplicación lineal, y en particular $F(X)$, puede representarse en su forma univariada de la siguiente manera:

$$F(X) = \sum_{i=0}^{n-1} A_i X^{q^i} + B,$$

donde A_i y B pertenecen a $\mathbb{K}$. Por lo que los sistemas de ecuaciones afines o las aplicaciones afines se pueden representar en su forma matricial, forma multivariable o forma univariada. Además de que se tenga una biyección entre dichas representaciones, se puede, de una manera eficiente, pasar de una a otra.

Proposición 4.2. *Dada una representación matricial de una aplicación afín $F(X)$, se puede obtener su representación univariada de una manera eficiente, y viceversa.*

Demostración. Solo hay que verlo en el caso de aplicaciones lineales, ya que el paso a aplicaciones afines se deduce del caso lineal inmediatamente. Siempre se puede suponer que el sistema tiene n ecuaciones, ya que, si faltan añadimos ceros y si sobran quitamos las linealmente dependientes. Como las aplicaciones de Frobenius son lineales se sabe que a cada una de ellas le corresponde una única matriz M_i. Dichas matrices forman base sobre $\mathbb{K}$ del espacio de aplicaciones lineales sobre $\mathbb{F}^n$. Además, a la aplicación lineal multiplicar por A_i le corresponde una única matriz $M(A_i)$. Por tanto, a la aplicación lineal $A_i X^{q^i}$ le

corresponde una única matriz $M(A_i)M_i$. Por tanto, se ha llegado a:

$$\begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n \end{pmatrix} = \sum_{i=0}^{n-1} M(A_i)M_i x.$$

Si fijamos un punto $x \in \mathbb{F}^n$, entonces en el lado izquierdo de la igualdad se tienen n^2 incógnitas a_{ij} y en el lado derecho se tienen n^2 incógnitas, que son las coordenadas de los elementos A_i. Esto quiere decir que si se conoce la representación matricial se conoce la parte izquierda y, por tanto, se tiene un sistema de n^2 incógnitas, que para poder resolver se tendrá que dar valores a x. Se tiene que dar n valores a x de manera que formen una base de $\mathbb{F}^n$, por ejemplo los e_i. De esta manera se llega a un sistema con n^2 incógnitas y n^2 ecuaciones, que tiene solución única ya que las aplicaciones de Frobenius forman una base sobre $\mathbb{K}$ del espacio de las aplicaciones lineales sobre $\mathbb{F}^n$. Resolviendo el sistema se obtiene la representación univariada de la aplicación F.

Análogamente, si se conoce la representación univariada se conoce la parte derecha de la igualdad, y de la misma manera que el paso anterior, se puede llegar a un sistema con solución única que da la representación matricial de la aplicación F. Hay que advertir que el paso de una representación a otra se hace en tiempo polinomial. □

Lo que se está haciendo en la demostración anterior es ver de una manera constructiva que se puede pasar de una representación a otra en un tiempo polinomial, ya que lo único que se tiene que hacer son operaciones básicas con polinomios y resolver sistemas lineales.

Veamos ahora la forma que tienen las aplicaciones polinomiales cuadráticas. Sea F una aplicación cuadrática de $\mathbb{F}^n$ a $\mathbb{F}^n$. Esto quiere decir que la componente k-ésima de F tiene la siguiente forma:

$$F_k(x) = \sum_{i,j=1}^{n} a_{ij}^k x_i x_j + \sum_{i=1}^{n} b_i^k x_i + c^k$$

donde los coeficientes a_{ij}^k, b_i^k y c^k pertenecen a $\mathbb{F}$. Veamos ahora como son los sistemas cuadráticos vistos desde $\mathbb{K}$.

Lema 4.3. *Considérese el siguiente tipo de aplicaciones sobre el cuerpo $\mathbb{K}$:*

$$P(X) = \sum_{i,j=0}^{n-1} A_{ij} X^{q^i+q^j} + \sum_{i=0}^{n-1} B_i X^{q^i} + C. \tag{4.1.1}$$

Se verifica que $\phi^{-1}(P(X)) \in \mathcal{MQ}(n,n)$.

Demostración. Análogo al caso anterior, si se utiliza el isomorfismo ϕ para ver como actúa la aplicación P sobre $\mathbb{F}^n$, se verá que el resultado es una aplicación cuadrática. Solo se tiene que ver que $X^{q^k+q^l}$ es una aplicación cuadrática, ya que como dijimos antes, los homomorfismos de Frobenius y multiplicar por un elemento de $\mathbb{K}$ son aplicaciones lineales. Si $(x_1, \ldots, x_n)$ son las coordenadas de X respecto de la base $\mathcal{B} = \{\beta_1, \ldots \beta_n\}$, entonces se tiene que:

$$\begin{aligned} X^{q^k+q^l} &= \left(\sum_{i=1}^{n} x_i\beta_i\right)^{q^k+q^l} = \left(\sum_{i=1}^{n} x_i\beta_i\right)^{q^k}\left(\sum_{i=1}^{n} x_i\beta_i\right)^{q^l} \\ &= \left(\sum_{i=1}^{n} h_i^1(x)\beta_i\right)\left(\sum_{i=1}^{n} h_i^2(x)\beta_i\right) \\ &= \sum_{i=1}^{n} h_i^3\left(\left(h_1^1(x), \ldots, h_n^1(x)\right), \left(h_1^2(x), \ldots, h_n^2(x)\right)\right)\beta_i \end{aligned}$$

donde h_i^1, h_i^2 son polinomios homogéneos de grado 1 y h_i^3 son polinomios homogéneos de grado 2. □

Proposición 4.4. *Fijado un isomorfismo $\phi : \mathbb{F}^n \to \mathbb{K}$, para cada $p \in \mathcal{MQ}(n, n)$ existe un único polinomio P de la misma forma que la ecuación 4.1.1 sobre $\mathbb{K}$, de manera que:*

$$P(X) = \phi\left(p\left(\phi^{-1}(X)\right)\right),$$

y viceversa.

Demostración. Sea $P(X)$ un polinomio de la misma forma que la ecuación 4.1.1, y sea Q el conjunto de todos los elementos con dicha forma. Por el lema 4.3, la aplicación ξ, definida por:

$$\begin{aligned} \xi : Q &\longrightarrow \mathcal{MQ}(n, n) \\ P &\longrightarrow p = \phi^{-1}(P(\phi(x))) \end{aligned}$$

está bien definida. Además, se tiene que es inyectiva. Para ello supongamos que $\xi(P_1) = \xi(P_2)$, entonces $P_1(X) = P_2(X)\ \forall X \in \mathbb{K}$ por lo que $P_1 = \lambda P_2$. Pero entonces tendríamos que $\xi(P_2) = \xi(P_1) = \xi(\lambda P_2) = \lambda\xi(P_2)$, por lo que $\lambda = 1$ y por tanto inyectiva. Ahora veamos que es suprayectiva, para lo cual se muestra que el número de elementos en $\mathcal{MQ}(n, n)$ es igual al número de elementos en Q. El número de coeficientes en un sistema de $\mathcal{MQ}(n, n)$ es $n\binom{n+2}{2}$ por lo que, el número de elementos distintos en $\mathcal{MQ}(n, n)$ es:

$$q^{n\binom{n+2}{2}}.$$

Una ecuación del tipo 4.1.1 tiene $\binom{n+2}{2}$ coeficientes en $\mathbb{K}$ por lo que el número de ecuaciones distintas es

$$q^{n\binom{n+2}{2}}.$$

Por tanto, la aplicación es suprayectiva por ser una aplicación inyectiva sobre conjuntos finitos del mismo tamaño. □

Observación 4.5. Dado un sistema de polinomios $p \in \mathcal{MQ}(n,n)$ se puede calcular de manera eficiente su ecuación univariada P y viceversa. La forma de hacerlo sería exactamente igual que en el caso de sistemas afines. Si se tiene P se calcula $\phi^{-1}(P(X))$. Si se tiene p se calcula para un polinomio genérico P el sistema de polinomios $\phi^{-1}(P(X))$, donde los coeficientes son indeterminadas. Dando valores a x y resolviendo los sistemas de ecuaciones lineales que resultan se obtiene el valor de los coeficientes. Dichos coeficientes dan el valor de P.

Observación 4.6. En caso de tener más o menos ecuaciones en un sistema de polinomios, lo único que habría que hacer sería añadir ecuaciones nulas o añadir más variables aunque no intervengan en el sistema. Así, se podrá hallar su representación univariada.

4.2. Sistema HFE

Veamos ahora un ejemplo fundamental de este tipo de sistemas. El sistema HFE fue desarrollado por Patarin en los años 90. La idea de este sistema es la misma que acabamos de ver en el apartado anterior, la dualidad que hay en ver un sistema de ecuaciones sobre el cuerpo $\mathbb{K}$ o sobre el $\mathbb{F}$-espacio vectorial de dimensión $[\mathbb{K}:\mathbb{F}]$. La aplicación de cifrado del HFE tendrá la siguiente forma:

$$F = L_1 \circ \phi^{-1} \circ \widetilde{F} \circ \phi \circ L_2.$$

Donde ϕ es la identificación entre $\mathbb{K}$ y $\mathbb{F}^n$, L_1 y L_2 aplicaciones afines invertibles y donde $\widetilde{F}$ es una aplicación de $\mathbb{K}$ en $\mathbb{K}$ que tiene la forma

$$F(X) = \sum_{i,j=0}^{r_1} A_{ij} X^{q^i+q^j} + \sum_{i=0}^{r_2} B_i X^{q^i} + C \tag{4.2.1}$$

con r_i números muy pequeños, de manera que $D < max(2q^{r_1}, q^{r_2})$ donde D es la cota para el sistema HFE. Esta cota D, es elegida por el usuario y ha de ser un valor suficientemente pequeño para poder invertir el sistema a través del algoritmo de Berlekamp.

Esta composición de funciones asegura un sistema polinomial cuadrático sobre $\mathbb{F}$. Cuando se quiere invertir esta función nos damos cuenta de que, en principio, no tendría porqué tener inversa, ya que la $\widetilde{F}$ no tiene porqué ser una biyección. Lo que sí se puede calcular, dado un valor y de la imagen, son todas sus preimágenes utilizando el algoritmo de Berlekamp.

El algoritmo de Berlekamp es uno de los mejores algoritmos para el calculo de soluciones de ecuaciones de una variable en cuerpos finitos. Para poder utilizar el algoritmo de Berlekamp es necesario que la ecuación 4.2.1 tenga grado bajo, ya que la complejidad del algoritmo de Berlekamp es $O\left(nd^2 \log d + d^3\right)$ [Sho09]. Por ejemplo, poniendo

$$2^{60} = (2q^i)^3$$

con $q = 2$ se tiene que $i \approx 19$. Por eso se pide que los r_i sean muy pequeños. Actualmente, este sistema no se considera seguro, ya que fue roto por Faugère y Joux [FJ03], mediante el algoritmo F4 desarrollado por Faugère. En principio, como la ecuación principal del sistema 4.2.1 es aleatoria, salvo la limitación del grado, cabría esperar que la utilización del F4 no daría resultado. En cambio, como se mostró en varios experimentos, el F4 era capaz de obtener una base de Groebner del sistema HFE en un tiempo razonable. Además, se vio que el grado máximo de los polinomios durante la ejecución del F4 era bajo (grados menores que 6), lo cual no es cierto para sistemas aleatorios en general.

El F4 es capaz de encontrar una base gracias a que hay ciertas relaciones polinomiales entre las ecuaciones del sistema, por lo que el grado de la base no se dispara durante la ejecución del algoritmo y hace que el first fall degree (primer grado en el que aparecen relaciones polinomiales no triviales entre los polinomios durante el algoritmo) sea pequeño. Esto es así por la limitación del grado en la ecuación 4.2.1, lo que produce que los sistemas no se comporten como sistemas aleatorios. Experimentalmente se ha comprobado que, cuando aparece una caída de grado, el algoritmo F4 encuentra una base en esa etapa del grado o en la siguiente. En cambio cuando trabajamos con sistemas aleatorios no es posible encontrar relaciones polinomiales no triviales de grado bajo.

Como se ve en la figura 4.2.1, si fijamos un D bajo, la complejidad del sistema no es exponencial, lo cual si pasa para D's elevados. La sección de la gráfica asociada a $\log D = 11$ se corresponde con el tiempo de ejecución de sistemas aleatorios, i.e, sin limitación en D. Por tanto, como se puede observar perfectamente a medida que la cota D aumenta, el sistema HFE se comporta como un sistema aleatorio y, por tanto, para D's altos la complejidad es exponencial.

Pero también hay que decir que la complejidad no depende exactamente del grado de las

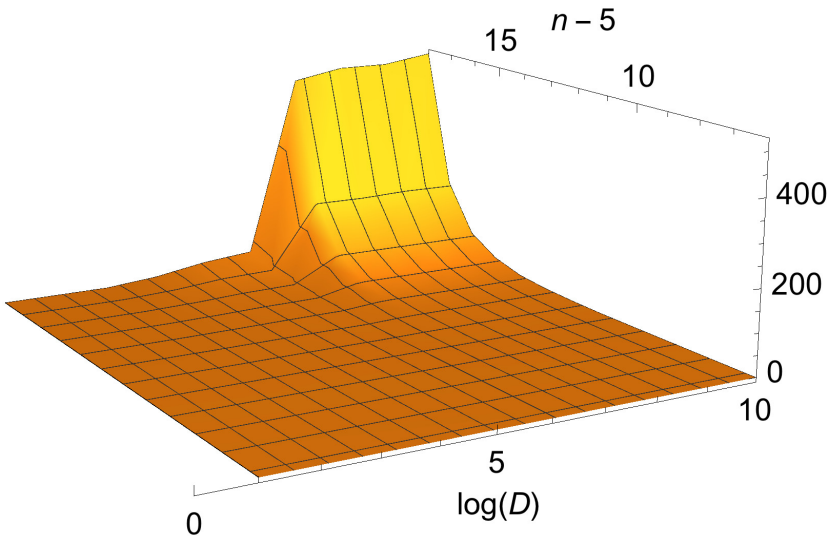

Figura 4.2.1: Tiempo necesario para resolver un sistema HFE

aplicaciones, sino de su grado 'intrínseco'. Veamos ahora cómo se puede definir el grado intrínseco. Sea $\mathcal{H}$ el conjunto de todas las aplicaciones univariadas sobre $\mathbb{K}$ del tipo HFE y $\mathcal{L}$ el conjunto de todas las aplicaciones univariadas sobre $\mathbb{K}$, que vistas sobre $\mathbb{F}$ son isomorfismos afines, y definamos la siguiente relación de equivalencia sobre $\mathcal{H}$:

$$F_1 \sim F_2 \iff \exists L_1, L_2 \in \mathcal{L} \mid F_1 = L_1 \circ F_2 \circ L_2$$

En esta situación sea $F \in \mathcal{H}$ y $\overline{F}$ su clase de equivalencia, definimos el grado intrínseco de F como

$$\deg^* F = \text{mín}\left\{\deg F' \mid F' \in \overline{F}\right\}$$

Normalmente, se habla del grado y no del grado intrínseco porque en la mayoría de los casos ambos grados coinciden. Por ejemplo, la complejidad de X^{q^2+q} es la misma que la de $X^{q^3+q^2}$ aunque ambas aplicaciones tengan grados distintos pero mismo grado intrínseco. Esto es así porque la complejidad de un sistema depende de la estructura interna del sistema que queda invariante por isomorfismos lineales, por lo que la complejidad queda invariante por isomorfismos lineales, lo cual se puede comprobar en los experimentos.

Como se ha visto antes, cualquier sistema cuadrático tiene asociado una ecuación univariada sobre $\mathbb{K}$ igual a la de los sistemas HFE. Cabría pensar que se puede utilizar el mismo proceso utilizado en el HFE para hallar las preimágenes de un sistema aleatorio cualquiera. Esto no es posible, ya que en general el grado de la ecuación univariada de un sistema

aleatorio es muy alto, de hecho lo normal es que tenga grado $q^{n-1} + q^{n-2}$ si la característica es 2. Lo cual hace que, para sistemas aleatorios, el algoritmo Berlekamp tenga una complejidad exponencial. De hecho, la idea de utilizar el proceso de desencriptado del HFE para resolver sistemas polinomiales es la que está detrás del algoritmo Zhuang-Zi.

4.3. Algoritmo Zhuang-Zi

Como ya se ha visto, hay varios métodos para resolver sistemas de ecuaciones polinomiales. Una de las formas de resolver estos sistemas es trasladar el sistema de polinomios sobre un cuerpo pequeño a una única ecuación sobre una extensión del cuerpo, para luego utilizar Berlekamp y obtener las soluciones. En esta sección se mostrará en detalle esta idea y se añadirán unas mejoras.

Sea f una aplicación cuadrática de $\mathbb{F}^n$ a $\mathbb{F}^n$, aunque se puede trabajar con aplicaciones polinomiales de cualquier orden. Por lo visto anteriormente, existe un único polinomio F sobre $\mathbb{K}$ que tiene la forma de la ecuación 4.1.1 de manera que:

$$F = \phi^{-1} \circ f \circ \phi.$$

Llegados a este punto el problema es que la aplicación F puede tener un grado elevado, ya que al ser f un sistema aleatorio el grado de F puede ser del orden de q^n. Por tanto, en general no se puede aplicarle a F el algoritmo de Berlekamp. La idea del Algoritmo Zhuang-Zi[Bar09] es buscar otro polinomio F' de menor grado y de manera que conociendo las soluciones de F' conozcamos las soluciones de F. La manera de proceder sería la siguiente:

1. Fíjese un grado máximo D para el cual el algoritmo de Berlekamp encuentra una solución en un tiempo razonable sobre el cuerpo $\mathbb{K}$. En el momento en el que cualquiera de los polinomios con los que se está trabajando tenga grado menor que D se pasa directamente al último paso.

2. Se consideran los homomorfismos de Frobenius

   $$G_i(X) = X^{q^i} \quad i = 0, \ldots, n-1.$$

   Y se calcula

   $$F_i(X) = G_i \circ F(X) = F^{q^i}(X) \quad \text{mód } (X^{q^n} - X)$$

Si nos damos cuenta, al componer con las aplicaciones de Frobenius, lo que se está haciendo es componer con una aplicación lineal, por lo tanto, conocidas las soluciones de F_i se podrá conocer las soluciones de F. Además, y aquí está la observación más importante, se está cambiado el grado máximo del polinomio.

3. Una vez calculados los F_i se considera la matriz de los coeficientes de los F_i. Aplicamos la reducción por filas y se obtiene así el conjunto $S = \{S_i(X) : 0 \leq i \leq t\}$. Hasta ahora solo se han realizado operaciones lineales en el cuerpo base.

4. Ahora para todo $i = 1, \ldots, t$ y $j = 1, \ldots, n-1$ se considera

$$X^{q^j} S_i(X) \mod (X^{q^n} - X)$$

y como antes, se considera la matriz de los coeficientes de los $X^{q^j} S_i(X)$. Se volverá a aplicar reducción por columnas y se obtiene el conjunto S'. Si existe algún elemento en S' con grado menor que D pasaríamos al siguiente paso, sino se repetiría sucesivamente este último paso renombrado S' por S.

5. Llegados a este punto se ha obtenido un $G(X)$ con grado menor que D. Ahora se calculan las raíces de G mediante Berlekamp y se obtiene así el conjunto

$$W = \{\alpha \mid G(\alpha) = 0\}$$

Las raíces de F serán el conjunto

$$W' = \{\alpha \in W \mid F(\alpha) = 0\}$$

Observación 4.7. En cualquier etapa, los ceros de $X^{q^j} S_i(X)$ o $F_i(X)$ contienen a los ceros de $F(X)$. Esto es así porque dichos polinomios se obtienen a partir de F por potencias, multiplicación por un monomio y suma entre ellos.

Observación 4.8. Detrás del cálculo de $X^{q^j} S_i(X) \mod (X^{q^n} - X)$ está la idea que subyace en el Algoritmo XL. Lo que se está haciendo es generar nuevas ecuaciones linealmente independientes con las que se puede obtener información del sistema y, además, las soluciones de nuestro sistema original están contenidas en las soluciones de dichas ecuaciones.

Observación 4.9. Este algoritmo también se puede aplicar en el caso en el que el número de ecuaciones sea distinto al número de variables. Si el número de ecuaciones es menor, al número de incógnitas, solo se tiene que añadir ecuaciones nulas. Si el número de ecuaciones,

es mayor al número de incógnitas, entonces se tiene que

$$m = nc + r$$

para ciertos c y r. Lo que se hará será separar nuestras ecuaciones en $c + 1$ grupos de n ecuaciones y calcular los correspondientes W_i' con $1 \leq i \leq c + 1$. Una vez hallados dichos conjuntos se tiene que las raíces de F son

$$W' = \bigcap_{i=1}^{c+1} W_i'$$

Ahora veamos una herramienta que he desarrollado para el análisis de sistemas MPKC.

4.4. Claves equivalentes

Cuando trabajamos con este tipo de sistemas puede darse el caso en el que dos claves privadas den la misma clave pública, este hecho se puede utilizar tanto para atacar ciertos sistemas como para mejorar los tamaños de las claves. Por tanto, se tiene que estudiar este tipo de situaciones, pero primero se presentará una visión general. Inicialmente el concepto de claves equivalentes en sistemas MPKC fue desarrollado en [WP], siendo utilizado para reducir el espacio de claves privadas y así poder reducir su tamaño en sistemas cuadráticos. En esta tesis se hace otro planteamiento distinto, el cual es utilizar el concepto de claves equivalentes para desarrollar un método de ataque general a sistemas MPKC, con el cual podemos hacer suposiciones sobre la clave privada y ayudarnos de la estructura del sistema para poder invertir la aplicación de cifrado. Veamos ahora dicho concepto.

Supongamos que el sistema se obtiene a partir de la composición de varias aplicaciones polinomiales, ya sean afines, cuadráticas o incluso de mayor grado. En este caso se tiene

$$P = F_s \circ \cdots \circ F_1$$

En principio, el usuario legítimo tiene la capacidad de invertir las F_i, por tanto, existe un algoritmo para invertir las F_i. Dicho algoritmo puede depender de ciertos parámetros involucrados en F_i. Si somos capaces de encontrar unas F_i' de manera que

$$P = F_s' \circ \cdots \circ F_1'$$

y que además las F_i' tengan la misma 'forma' que las F_i entonces se podrá resolver el

sistema. Con misma forma se quiere decir que el algoritmo para invertir F_i sea también válido para invertir F_i'. En este caso, se dirá que los sistemas están relacionados

$$(F_s, \ldots, F_1) \approx \left(F_s', \ldots, F_1'\right)$$

Claramente $\approx$ es una relación de equivalencia, por tanto, esta técnica también se puede utilizar para reducir el tamaño de la clave. Esto es así ya que siempre se podrá definir una clave normalizada de cada clase generada por la relación $\approx$, y, por tanto, reducir el número de claves redundantes.

El generar sistemas relacionados no es algo sencillo, pero nos podremos restringir a sistemas equivalentes que tengan una forma determina para así poder hallarlos. Lo que se tiene que hacer es buscar aplicaciones θ_i de manera que $\theta_i \circ F_i \circ \theta_{i-1}^{-1}$ tenga la misma 'forma' que F_i. Una vez encontradas, operamos y se obtiene

$$P = F_s \circ \cdots \circ F_1 = \left(F_s \circ \theta_{s-1}^{-1}\right) \circ \cdots \circ \left(\theta_2 \circ F_2 \circ \theta_1^{-1}\right) \circ (\theta_i \circ F_1)$$

De esta manera considerando

$$F_i' = \left(\theta_i \circ F_i \circ \theta_{i-1}^{-1}\right)$$

se obtiene un sistema equivalente.

Simplificando un poco, en el caso genérico de los sistemas MPKC, se deben encontrar unas θ_1 y θ_2 lineales (o afines) de manera que $\theta_2^{-1} \circ F \circ \theta_1$ tenga la misma forma que F. Además, será útil considerar que $T \circ \theta_2$ y $\theta_1^{-1} \circ S$ cumplan unas determinadas condiciones para que puedan ser halladas. Este procedimiento se puede hacer mucho más general, teniendo siempre en mente la idea de buscar otra clave privada válida que dé la misma clave pública. Si se supone que el sistema tiene la forma anterior, entonces se puede formalizar la definición de la siguiente manera.

Definición 4.10. Fijado el tipo sistema criptográfico MPKC definimos el espacio de todas las posibles claves privadas $\mathcal{S} = \{(T, F, S)\}$, donde F puede ser composición de varias aplicaciones polinomiales, y el espacio de todas las posibles claves públicas $\mathcal{P} = \{P\}$. Se dirá que dos sistemas están relacionados

$$(T, F, S) \approx \left(T', F', S'\right)$$

si sus respectivas claves públicas son iguales, $P = P'$.

Esta definición es un poco más débil que la presentada anteriormente, ya que no se impone

que $F' = \theta_2^{-1} \circ F \circ \theta_1$. Además, esta definición es más manejable en la práctica, ya que deja mayor libertad en la búsqueda de sistemas relacionados. Claramente se tiene que:

$$\mathcal{P} = \mathcal{S}/\approx$$

En la práctica para poder encontrar claves equivalentes útiles que den cierta información de la clave privada, lo primero que se tiene que hacer es encontrar el espacio de las transformaciones θ_1 y θ_2 (lineales o afines) que dejen los sistemas invariantes. Una vez hallado dicho espacio se ve que tipo de simplificaciones o suposiciones se pueden hacer sobre T, F y S a través de las transformaciones θ_1 y θ_2. Con todo esto se llega a un sistema (T', F', S') con clave pública P y del que, además, se conoce cierta información sobre la clave privada. Como ya se ha comentado, puedan darse más situaciones generales en las que los sistemas no tengan esta forma determinada o se pueda considerar transformaciones θ_i no lineales.

Ejemplo 4.11 (HFE)**.** En el HFE, si se considera θ_1 y θ_2 como las transformaciones lineales asociadas a las aplicaciones sobre $\mathbb{F}_{q^n}$ de la forma

$$\sum_i a_i X^{q^{\alpha_i}}$$

con los q^{α_i} pequeños, entonces se tiene que dichas transformaciones dejan invariantes los sistemas HFE. Ya que, como los q^{α_i} son pequeños, no se está aumentado mucho el grado de la ecuación principal del sistema HFE y, por tanto, se podrá invertir utilizando Berlekamp. Se pueden dar más ejemplos de algunas de las transformaciones que pueden utilizarse en determinados sistemas MPKC.

Capítulo 5

Criptosistema ME

En este capítulo vamos a presentar y analizar el sistema multivariable ME (Matrix Exponentiation) propuesto por Ignacio Luengo, basado en fewnomials sobre el cuerpo $\mathbb{F}_q$, donde $q = 2^p$. A diferencia de los demás sistemas multivariables, las ecuaciones de este sistema no son cuadráticas, sino que los exponentes son del orden de q. En cambio, el número de monomios por ecuación será fijo y de orden bajo, por ejemplo de la manera que se define en este capítulo el número de monomios es 9.

La idea del sistema es combinar un automorfismo lineal, un endomorfismo monomial de grado alto y un endomorfismo triangular. Como luego se verá los endomorfismos serán inyectivos, si nos restringimos a un subconjunto de $\mathbb{F}_q^m$. La idea fundamental del morfismo triangular es la misma que se utiliza en los sistemas propuestos por T. T. Moh llamados TTM [Moh99] o más en general los sistemas triangulares TPM [DGS06]. La idea es considerar una aplicación polinomial de manera que, dado un punto cualquiera de la imagen se pueda hallar de forma eficiente su preimagen a través de un procedimiento recursivo, aunque el cálculo explícito de la fórmula de la aplicación inversa no se pueda hacer en tiempo polinomial.

Notación 5.1. Utilizamos la siguiente notación.

1. Sea $A \in \mathcal{M}_{m\times m}(\mathbb{Z}_{q-1})$, la fila i-ésima de A la se denotará por $A(i)$.
2. Si $a \in \mathbb{Z}_{q-1}^m$ y $v \in \mathbb{F}_q^m$ definimos $v^a := v_1^{a_1} v_2^{a_2} \dots v_m^{a_m}$.

3. Sea $u \in \mathbb{F}_q^m$ y $A \in \mathcal{M}_{m\times m}(\mathbb{Z}_{q-1})$, definimos

$$A * u = A * \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} := \begin{pmatrix} u_1^{a_{11}} \cdot u_2^{a_{12}} \cdot \cdots \cdot u_m^{a_{1m}} \\ \vdots \\ u_1^{a_{11}} \cdot u_2^{a_{12}} \cdot \cdots \cdot u_m^{a_{1m}} \end{pmatrix} = \begin{pmatrix} u^{A(1)} \\ \vdots \\ u^{A(m)} \end{pmatrix}$$

4. Sea $y \in \mathbb{F}_q^m$, definimos $\widetilde{y}_r := (y_{r+1}, \ldots, y_m)$.

5. Los vectores se considerarán como vectores fila o columna indistintamente para mejorar la lectura, a menos que se indique explícitamente. Siempre hay que considerar la elección más natural.

5.1. Construcción del sistema

Ahora definimos las aplicaciones involucradas en nuestro sistema. Normalmente los sistemas multivariables son de la forma

$$P = T \circ Q \circ S$$

donde T y S son automorfismos lineales y Q es un monomorfismo polinomial cuadrático. El esquema general se muestra en la figura 5.1.1, donde la aplicación P es la clave pública con la que cualquier persona puede cifrar un texto x sin más que evaluar $P(x)$.

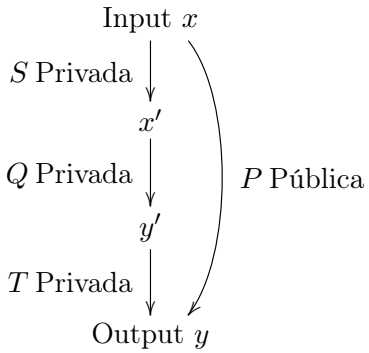

Figura 5.1.1: Esquema general de los MPKC.

Como ya se ha comentado, nuestro sistema no sigue el esquema general. Sea $P : \mathbb{F}^m \to \mathbb{F}^m$ la aplicación de cifrado, donde $\mathbb{F} = \mathbb{F}_{2^p}$ y $P = P_1 \circ Q_1 \circ P_0$. Veamos cómo se tiene que definir P_1, Q_1 y P_0 para generar el sistema.

1. La primera aplicación, P_0, actúa como una aplicación lineal sobre los exponentes. Definimos

$$P_0(x) = A * x = u$$

donde $A \in \mathcal{M}_{m\times m}(\mathbb{Z}_{q-1})$ es una matriz aleatoria con $det(A)$ coprimo con $q-1$. La condición sobre el $\det(A)$ es necesaria, como luego se verá, para poder invertir la aplicación sobre cierto subconjunto de $\mathbb{F}_q^m$.

2. La idea de la aplicación Q_1 es muy parecida a las de las aplicaciones utilizadas en los sistemas TTM. Definimos $Q_1(u) = y$ de la siguiente manera:

$$y = Q_1(u) = \begin{cases} y_1 = \widetilde{u}_1^{\mu_1^1} u_1 + \widetilde{u}_1^{\mu_1^2} + \widetilde{u}_1^{\mu_1^3} \\ \vdots \\ y_j = \widetilde{u}_j^{\mu_j^1} u_j + \widetilde{u}_j^{\mu_j^2} + \widetilde{u}_j^{\mu_j^3} \\ \vdots \\ y_n = \widetilde{u}_n^{\mu_n^1} u_n + \widetilde{u}_n^{\mu_n^2} + \widetilde{u}_n^{\mu_n^3} \\ y_{n+1} = \widetilde{u}_n^{\mu_{n+1}} \\ \vdots \\ y_m = \widetilde{u}_n^{\mu_m} \end{cases}$$

Con los $\mu_i^j \in \mathbb{Z}_{q-1}^{m-i}$ y $\mu_i \in \mathbb{Z}_{q-1}^{m-n}$ elementos aleatorios. Además, se verifica que

$$\dim\left(L\left[\mu_{n+1}, \ldots, \mu_m\right]\right) = m - n.$$

Para simplificar la notación renombramos $\mu_j^1 + e_j$ por μ_j^1, con lo que $\widetilde{u}_j^{\mu_j^1} u_j$ pasaría a

ser $\widetilde{u}_j^{\mu_j^1}$. Si se considera $y = Q_1 \circ P_0(x)$ se obtienen las ecuaciones

$$y = Q_1 \circ P_0(x) = \begin{cases} y_1 = x^{M_1^1} + x^{M_1^2} + x^{M_1^3} \\ \vdots \\ y_j = x^{M_j^1} + x^{M_j^2} + x^{M_j^3} \\ \vdots \\ y_n = x^{M_n^1} + x^{M_n^2} + x^{M_n^3} \\ y_{n+1} = x^{M_{n+1}} \\ \vdots \\ y_m = x^{M_m} \end{cases}$$

Donde M_k^j y M_k son de la forma $\mu_k^j A$ y $\mu_k A$ respectivamente.

3. La última aplicación es muy parecida a la primera pero con algunas restricciones. Definimos $P_1(y) = C * y = z$, donde:

$$C = \left(\begin{array}{ccc|ccc} \lambda_{11} & \dots & \lambda_{1n} & \lambda_{1n+1} & \dots & \lambda_{1m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1} & \dots & \lambda_{mn} & \lambda_{mn+1} & \dots & \lambda_{mm} \end{array} \right) \tag{5.1.1}$$

Veamos qué propiedades cumple la matriz C. Fijada la fila j se ha de cumplir que en la parte izquierda de la matriz solo haya dos entradas no nulas λ_{jk_1} y λ_{jk_2}, que serán de la forma $2^{\beta_{jk_1}}$ y $2^{\beta_{jk_2}}$ (evidentemente k_1 y k_2 dependen de j). Para abreviar, dichas entradas no nulas se denotarán como a_j y b_j respectivamente. Pedimos que haya solo dos entradas no nulas por fila para que el número de monomios sea exactamente 9.

Estas son todas las aplicaciones necesarias para formar nuestro sistema quedando como clave pública:

$$P(x) = P_1 \circ Q_1 \circ P_0(x) = (z_1, \dots, z_m)$$

Veamos ahora que forma tiene nuestro sistema. Para un j fijado componiendo las 3 apli-

caciones se obtiene:

$$\begin{aligned} z_j &= y_{k_1}^{a_j} \cdot y_{k_2}^{b_j} \cdot \widetilde{y_n^{C(j)_n}} \\ &= \left(x^{a_j M_{k_1}^1} + x^{a_j M_{k_1}^2} + x^{a_j M_{k_1}^3}\right) \cdot \left(x^{b_j M_{k_2}^1} + x^{b_j M_{k_2}^2} + x^{b_j M_{k_2}^3}\right) \cdot x^{N_j} \\ &= \sum_{k,l=1}^{3} x^{H_j^{kl}} \end{aligned}$$

Donde

$$H_j^{kl} = a_j M_{k_1}^k + b_j M_{k_2}^l + N_j$$

y

$$\widetilde{y_n^{C(j)_n}} = \prod_{k=n+1}^{m} \left(x^{M_k}\right)^{C(j)_k} = x^{\sum_{k=n+1}^{m} M_k \cdot C(j)_k} = x^{N_j}$$

Veamos ahora ciertos elementos que nos servirán de utilidad más adelante. Definimos:

$$\begin{cases} v_1 = a_j M_{k_1}^1 = a_j \mu_{k_1}^1 A \\ v_2 = a_j M_{k_1}^2 = a_j \mu_{k_1}^2 A \\ v_3 = a_j M_{k_1}^3 = a_j \mu_{k_1}^3 A \\ v_4 = b_j M_{k_2}^1 = b_j \mu_{k_2}^1 A \\ v_5 = b_j M_{k_2}^2 = b_j \mu_{k_2}^2 A \\ v_6 = b_j M_{k_2}^3 = b_j \mu_{k_2}^3 A \\ v_7 = N_j = \left(\sum_{k=n+1}^{m} C(j)_k \mu_k\right) A \end{cases}$$

Para un j fijo definimos la matriz H_j como la matriz de dimensión $9 \times m$ de manera que:

$$H_j = \begin{pmatrix} H_j^{11} \\ H_j^{12} \\ H_j^{13} \\ H_j^{21} \\ H_j^{22} \\ H_j^{23} \\ H_j^{31} \\ H_j^{32} \\ H_j^{33} \end{pmatrix} = \begin{pmatrix} v_1 + v_4 + v_7 \\ v_1 + v_5 + v_7 \\ v_1 + v_6 + v_7 \\ v_2 + v_4 + v_7 \\ v_2 + v_5 + v_7 \\ v_2 + v_6 + v_7 \\ v_3 + v_4 + v_7 \\ v_3 + v_5 + v_7 \\ v_3 + v_6 + v_7 \end{pmatrix}$$

Los H_j^{kl} son la clave pública. Cuando fijamos uno o más índices, por ejemplo H^{11}, significa que se está considerando una matriz cuyos elementos son los H_j^{kl} que comparten dichos índices fijados.

Se tiene que $u = A * x$, por tanto si quisiéramos obtener las ecuaciones en función de las $u's$ se tiene que:

$$z_j = \sum_{k,l=1}^{3} u^{\bar{H}_j^{kl}}$$

donde la matriz $\bar{H}_j$ verifica que:

$$H_j^{kl} = \sum_{r=1}^{m} \bar{H}_j^{kl}(r) \cdot A(r)$$

Por tanto, $\bar{H}_j = H_j \cdot A^{-1}$.

Observación 5.2. Se puede cambiar el sistema multiplicando a la derecha por otra aplicación de la forma $x = R * w$, dando lugar a la aplicación $\psi = P_1 \circ Q_1 \circ P_0 \circ R(w)$. Con esto se podría simplificar los exponentes de n términos en la clave pública, solo se tiene que tomar R^{-1} como la matriz cuya fila i es el exponente de un monomio de la ecuación i para todo $1 \leq i \leq n$; con esto conseguimos que al hacer la sustitución $x = R * w$ los monomios elegidos tengan exponente 1. Este hecho no pone en riesgo la seguridad del sistema. Aunque, si se supiera una solución de $z = \psi(w)$ entonces se puede resolver nuestro sistema inicial tomando $x = R * w$.

Veamos ahora una proposición importante que será útil después.

Proposición 5.3. *Sea $z = B * (A * x)$, entonces si $z_i = x^{\lambda_i}$ se tiene que:*

$$\lambda_i = \sum_{j=1}^{n} B(i)(j)A(j)$$

Por tanto,

$$z = B * (A * x) = (BA) * x. \tag{5.1.2}$$

Demostración. Solo hay que calcular z a partir de la definición de $*$. □

Observación 5.4. Como se está en el cuerpo finito de cardinal q los exponentes de una variable nunca son mayores o iguales a q, ya que las soluciones solo nos interesan en $\mathbb{F}_q$. Además, si se hiciera un cambio de variable $y = A * x$, se puede volver a hacer otro cambio

de variable de la forma $z = B * y$ de manera que $z = x$; siempre que nos restrinjamos al subconjunto

$$W = \{x \mid x_i \neq 0 \, \forall i\}$$

y que el determinante de la matriz A sea coprimo con $q-1$. Nos restringimos al subconjunto W ya que en otro caso el primer cambio de variable no tendría por que ser inyectivo en el caso general, ya que los exponentes no viven exactamente en $\mathbb{Z}_{q-1}$ ya que

$$x_i^{q-1} \neq x_i^0 = 1$$

a no ser que $x_i \neq 0$. Por tanto, si nos restringimos al conjunto W los exponentes pertenecen a $\mathbb{Z}_{q-1}$, con lo que tomando una matriz A con $det(A)$ coprimo con $q-1$ y por la proposición 5.3 se tiene que

$$x = A^{-1} * (A * x)$$

Ejemplo 5.5. Si tomamos la matriz A que sea triangular superior con todas las entradas 1's se tiene que la matriz es invertible mód $q-1$ pero la aplicación resultante es de la forma $(y_1, \ldots, y_n) = (x_1 \ldots x_n, x_2 \ldots x_n, \ldots, x_n)$ la cual no es inyectiva y, por tanto, no es invertible. A no ser, como ya se ha mencionado, que nos restrinjamos al subespacio W.

5.2. Relaciones

En este apartado vamos a presentar y definir ciertas relaciones que se dan entre los elementos de la clave pública, que a priori no dicen nada pero luego serán de utilidad. Como luego se verá se pueden obtener las columnas de la parte izquierda de la matriz C, aunque en principio no se puede saber sus posiciones. Fijado j se tiene que:

$$\begin{aligned}
V_{21}^j &:= H_j^{21} - H_j^{11} = v_2 - v_1 = a_j \left(M_{k_1}^2 - M_{k_1}^1\right) = a_j(\mu_{k_1}^2 - \mu_{k_1}^1)A = a_j(0, \ldots, 0, 1, *, \ldots, *)A \\
V_{31}^j &:= H_j^{31} - H_j^{11} = v_3 - v_1 = a_j \left(M_{k_1}^3 - M_{k_1}^1\right) = a_j(\mu_{k_1}^3 - \mu_{k_1}^1)A = a_j(0, \ldots, 0, 1, *, \ldots, *)A \\
V_{32}^j &:= H_j^{31} - H_j^{21} = v_3 - v_2 = a_j \left(M_{k_1}^3 - M_{k_1}^2\right) = a_j(\mu_{k_1}^2 - \mu_{k_1}^2)A = a_j(0, \ldots, 0, 1, *, \ldots, *)A \\
V_{54}^j &:= H_j^{12} - H_j^{11} = v_5 - v_4 = b_j \left(M_{k_2}^2 - M_{k_2}^1\right) = b_j(\mu_{k_2}^2 - \mu_{k_2}^1)A = b_j(0, \ldots, 0, 1, *, \ldots, *)A \\
V_{64}^j &:= H_j^{13} - H_j^{11} = v_6 - v_4 = b_j \left(M_{k_2}^3 - M_{k_2}^1\right) = b_j(\mu_{k_2}^3 - \mu_{k_2}^1)A = b_j(0, \ldots, 0, 1, *, \ldots, *)A \\
V_{65}^j &:= H_j^{13} - H_j^{12} = v_6 - v_5 = b_j \left(M_{k_2}^3 - M_{k_2}^2\right) = b_j(\mu_{k_2}^3 - \mu_{k_2}^2)A = b_j(0, \ldots, 0, 1, *, \ldots, *)A
\end{aligned}$$

Como adelanto, si dos filas de la matriz C verifican que tienen un elemento no nulo para una cierta columna entonces se verifica que varios elementos V_{**}^j son proporcionales a varios $V_{**}^{j'}$.

Por las relaciones anteriores se tiene que:

$$L[\{H_j^{kl}\}] = L[H_j^{11}, V_{21}^j, V_{31}^j, V_{54}^j, V_{64}^j]$$

Por otra parte, suponiendo que se conoce los valores k_1 y k_2 (como luego se verá, se puede suponer sin pérdida de generalidad), se tiene que

$$\begin{aligned}
a_j^{-1}V_{21}^j &= (\mu_{k_1}^2 - \mu_{k_1}^1)A = (0,\ldots,0,1,*,\ldots,*)A \\
a_j^{-1}V_{31}^j &= (\mu_{k_1}^3 - \mu_{k_1}^1)A = (0,\ldots,0,1,*,\ldots,*)A \\
a_j^{-1}V_{32}^j &= (\mu_{k_1}^3 - \mu_{k_1}^2)A = (0,\ldots,0,1,*,\ldots,*)A \\
b_j^{-1}V_{54}^j &= (\mu_{k_2}^2 - \mu_{k_2}^1)A = (0,\ldots,0,1,*,\ldots,*)A \\
b_j^{-1}V_{64}^j &= (\mu_{k_2}^3 - \mu_{k_2}^1)A = (0,\ldots,0,1,*,\ldots,*)A \\
b_j^{-1}V_{65}^j &= (\mu_{k_2}^3 - \mu_{k_2}^2)A = (0,\ldots,0,1,*,\ldots,*)A
\end{aligned}$$

de donde se puede obtener unas $2nm$ ecuaciones cuadráticas. También se tiene que

$$C\begin{pmatrix} \vdots \\ M_j^k \\ \vdots \\ M_j \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ H_j^{kk} \\ \vdots \end{pmatrix} = H^{kk} \quad k = 1,2,3$$

Por ejemplo, en el caso de $k = 1$ se obtiene

$$C\begin{pmatrix} M_1^1 \\ M_2^1 \\ \vdots \\ M_n^1 \\ M_{n+1} \\ \vdots \\ M_m \end{pmatrix} = \begin{pmatrix} H_1^{11} \\ H_2^{11} \\ \vdots \\ H_m^{11} \end{pmatrix}$$

5.3. Descifrado

En este apartado, nos centramos en como se puede descifrar si somos un usuario legítimo. Además, se presentarán ciertas propiedades que cumple el sistema, las cuales se utilizarán para su análisis. Hay que tener en cuenta que para invertir la aplicación $*$ es necesario que

x tenga todas sus coordenadas no nulas, como ya se ha comentado anteriormente.

Sea z_0 un texto cifrado. Los pasos a seguir para descifrar un texto cifrado son los siguientes:

1. Lo primero que se tiene que hacer es invertir la aplicación P_1, para ello se tiene que calcular la inversa de C módulo $p-1$, la cual se sabe que existe. Una vez hallada la inversa se calcula $y_0 = C^{-1} * z_0$.

2. En este punto se llega a un sistema de la forma

$$\begin{cases} y_1 = \widetilde{u}_1^{\mu_1^1} u_1 + \widetilde{u}_1^{\mu_1^2} + \widetilde{u}_1^{\mu_1^3} \\ \vdots \\ y_j = \widetilde{u}_j^{\mu_j^1} u_j + \widetilde{u}_j^{\mu_j^2} + \widetilde{u}_j^{\mu_j^3} \\ \vdots \\ y_n = \widetilde{u}_n^{\mu_n^1} u_n + \widetilde{u}_n^{\mu_n^2} + \widetilde{u}_n^{\mu_n^3} \\ y_{n+1} = \widetilde{u}_n^{\mu_{n+1}} \\ \vdots \\ y_m = \widetilde{u}_n^{\mu_m} \end{cases}$$

   Como los μ_j's tienen dimensión $m-n$, se puede generar la matriz $B = (\mu_{n+1}, \ldots, \mu_m)^t$, hallar su inversa módulo $q-1$ y calcular, igual que antes,

$$\begin{pmatrix} u_{n+1} \\ \vdots \\ u_m \end{pmatrix} = B^{-1} * \begin{pmatrix} y_{n+1} \\ \vdots \\ y_m \end{pmatrix}.$$

   Llegados a este punto se puede calcular u_n, ya que

$$u_n = \frac{y_n - \widetilde{u}_n^{\mu_n^2} - \widetilde{u}_n^{\mu_n^3}}{\widetilde{u}_n^{\mu_n^1}}. \tag{5.3.1}$$

   Los elementos μ_j^k's y μ_j's verifican una propiedad importante con la cual se puede invertir el sistema. Es una de las ideas fundamentales, muy similar a la idea que tiene el criptosistema TTM.

   Definición 5.6. Se dirá que los elementos

$$\left\{ v_j^k, v_{j'} : 1 \leq k \leq 3, 1 \leq j \leq n, n+1 \leq j' \leq m \right\}$$

verifican la estructura triangular si:

- Los elementos
$$\left\{v_{j'}, v_j^1 : 1 \leq j \leq n, n+1 \leq j' \leq m\right\}$$
son linealmente independientes.
- Los vectores v_l^k verifican que
$$v_l^k \in L\left[\left\{v_{j'}, v_j^1 : l+1 \leq j \leq n, n+1 \leq j' \leq m\right\}\right]$$
para $k = 2, 3$.

Es evidente por la definición que los μ_j^k y μ_j verifican la estructura triangular. Por tanto, análogamente a la ecuación 5.3.1 se procederá de manera recursiva para obtener

$$u_j = \frac{y_j - \widetilde{u}_j^{\mu_j^2} - \widetilde{u}_j^{\mu_j^3}}{\widetilde{u}_j^{\mu_j^1}}$$

para todo $1 \leq j \leq n$. De esta manera se obtiene u.

3. Finalmente se obtiene $x = A^{-1} * u$.

En principio, este sería el método empleado para descifrar un mensaje, pero en realidad no es necesario tener toda la clave privada para descifrar un mensaje. Es suficiente con conocer los valores de C y de los M_j^k's y M_j's para descifrar los mensajes. La manera de proceder sería la siguiente:

1. Igual que antes se calcula $y_0 = C^{-1} * z_0$.
2. El segundo paso sería invertir directamente $Q_1 \circ P_0$. Esto es posible porque la estructura triangular de los u_j^k's y u_j's la heredan los M_j^k's (salvo permutación en los subíndices) y M_j's, ya que A tiene rango máximo. Por tanto, se sabe que existe j_1 tal que $M_{j_1}^2$ y $M_{j_1}^3$ pertenezcan al espacio
$$L\left[\{M_j : n+1 \leq j \leq m\}\right].$$
Esto quiere decir que $M_{j_1}^2$ y $M_{j_1}^3$ son una combinación lineal de los M_j's y, por tanto, $x^{M_{j_1}^2}$ y $x^{M_{j_2}^2}$ se puede poner como una expresión polinomial en los
$$\left\{x^{M_j} : n+1 \leq j \leq m\right\}$$

y así conseguimos hallar los valores de $x^{M^2_{j_1}}$ y $x^{M^2_{j_2}}$. Una vez obtenidos dichos valores se obtiene $x^{M^1_{j_1}}$. Ahora se tiene que buscar un j_2 de manera que $M^2_{j_2}$ y $M^3_{j_2}$ pertenezcan al espacio $L\left[\{M_j : n+1 \leq j \leq m\} \cup \left\{M^1_{j_0}\right\}\right]$ y se procedería de la misma manera para hallar $x^{M^1_{j_2}}$. Repitiendo el proceso se llega a un sistema que tiene la forma:

$$\begin{cases} a_1 = x^{M^1_1} \\ \vdots \\ a_j = x^{M^1_j} \\ \vdots \\ a_n = x^{M^1_n} \\ a_{n+1} = x^{M_{n+1}} \\ \vdots \\ a_m = x^{M_m} \end{cases}$$

donde los a_j's son conocidos. Como se sabe que la matriz

$$\mathfrak{M} = \left(M^1_1, \ldots, M^1_n, M_{n+1}, \ldots, M_m\right)^t$$

tiene rango máximo, se puede invertir módulo $q-1$.

3. Finalmente se obtiene $x = \mathfrak{M}^{-1} * a$.

5.4. Análisis del sistema

En esta sección se procederá al análisis del sistema con el método de las claves equivalentes. Con este método se puede romper el sistema solo conociendo la clave pública. Por este motivo el sistema no se ha patentado, aunque el estudio ha ayudado a encontrar otras variantes más resistentes, como por ejemplo el sistema DME ya patentado presentado al NIST y en proceso de revisión como estándar en una era post-cuántica. La idea es la siguiente.

5.4.1. Claves equivalentes

Retomemos la idea de claves equivalentes y veamos como podemos aplicarlo a nuestro sistema.

Sean $\mathcal{S}$ y $\mathcal{P}$ el conjunto de todas las posibles claves privadas y públicas del sistema multivariable que vamos a estudiar. Para afinar un poco más se puede pensar en $\mathcal{P}$ módulo transformaciones lineales, ya que en la práctica se puede discernir de manera eficiente si dos sistemas están relacionados por una transformación lineal. Está claro que un sistema estará unívocamente determinado por su clave privada S y a cada clave privada S le corresponde una única clave pública. La clave pública de un sistema multivariable es un sistema de ecuaciones polinomiales, mientras que la clave privada es un conjunto de aplicaciones de manera que al componerlas den la clave pública. Definimos la aplicación

$$\Theta : \mathcal{S} \to \mathcal{P}$$

que manda cada clave privada a su clave pública asociada. Está claro que la función Θ está bien definida, es suprayectiva y en general no es inyectiva. Por tanto, automáticamente surge pensar en la relación de equivalencia

$$S \sim S' \iff \Theta(S) = \Theta(S')$$

que relaciona claves privadas si tienen la misma clave pública, obteniendo

$$\mathcal{P} = \mathcal{S}/\sim$$

Supongamos que se quiere analizar un sistema I con clave privada S del cual no se conoce nada salvo su clave pública, y supongamos que, de alguna manera, se puede encontrar un sistema I' con clave privada S' que esté en la misma clase de equivalencia que S y del cual se conoce algún dato de la clave privada, lo cual facilitaría el análisis. En ese caso, sin pérdida de generalidad se puede suponer que se está analizando el sistema I', ya que si conseguimos descifrar un mensaje total o parcialmente se podrá hacer lo mismo para cualquier sistema cuya clave privada esté en la clase de equivalencia de S' y en particular para I.

Este planteamiento teórico, en la práctica no se puede aplicar directamente, ya que no se tienen las clases de equivalencia, por lo que se tiene que encontrar los elementos relacionados ad-hoc. En la práctica, lo que se tendrá que hacer será encontrar un sistema equivalente al de partida, i.e, que tenga la misma clave pública. Si se quiere buscar un sistema equivalente al sistema

$$I = \{S, P\} = \{\{S_1, \dots S_s\}, P\}$$

lo que se tendrá que hacer será buscar un conjunto de transformaciones

$$\Phi = \{\phi_1, \ldots, \phi_s\}$$

de manera que

$$\Theta(S') = \Theta(\{\phi_i(S_i)\}_{i=1,\ldots s}) = P$$

y, además, que el conjunto $\{\phi_i(S_i)\}_{i=1,\ldots s}$ sea la clave privada de un sistema, o al menos que cumpla las propiedades necesarias para poder invertirlo.

5.4.2. Consideraciones previas

En esta subsección se verán ciertas propiedades que verifican nuestro sistema, las cuales se utilizarán más tarde para su análisis.

Sea

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

Lo primero será determinar los elementos de la parte izquierda de C. Para ello vamos a calcular los cocientes entre los elementos no nulos de una misma columna de la parte izquierda de C. Esto es posible comparando las diferencias V_{kl}^j's de la clave pública. Supongamos que dos elementos de una misma columna k (la cual no se conoce) son no nulos y, además, dichos elementos están en las filas j_1 y j_2 respectivamente, por tanto dichos elementos serán $\lambda_{j_1 k}$ y $\lambda_{j_2 k}$. Se debe tener en cuenta la posición relativa que ocupa cada elemento en su fila, ya que dependiendo de en que posición se encuentre cada elemento habrá que considerar distintos vectores. Por tanto, definimos:

$$\epsilon_i = 0 \text{ si } \lambda_{j_i k} \text{ es el primer elemento no nulo de su fila y } \epsilon_i = 1 \text{ en otro caso.}$$

Por tanto, como se ha supuesto que dichos elementos comparten columna necesariamente existen μ_1, μ_2 y μ_3 enteros positivos tales que:

- Si $\epsilon_1 = \epsilon_2 = 0$ entonces

$$V_{21}^{j_1} = 2^{\mu_1} V_{21}^{j_2},\ V_{31}^{j_1} = 2^{\mu_2} V_{31}^{j_2} \text{ y } V_{32}^{j_1} = 2^{\mu_3} V_{32}^{j_2}$$

- Si $\epsilon_1 = \epsilon_2 = 1$ entonces

$$V_{54}^{j_1} = 2^{\mu_1} V_{54}^{j_2},\ V_{64}^{j_1} = 2^{\mu_2} V_{64}^{j_2} \text{ y } V_{65}^{j_1} = 2^{\mu_3} V_{65}^{j_2}$$

- Si $\epsilon_1 = 0$ y $\epsilon_2 = 1$ entonces

$$V_{21}^{j_1} = 2^{\mu_1} V_{54}^{j_2},\ V_{31}^{j_1} = 2^{\mu_2} V_{64}^{j_2} \text{ y } V_{32}^{j_1} = 2^{\mu_3} V_{65}^{j_2}$$

- Si $\epsilon_1 = 1$ y $\epsilon_2 = 0$ entonces

$$V_{54}^{j_1} = 2^{\mu_1} V_{21}^{j_2},\ V_{64}^{j_1} = 2^{\mu_2} V_{31}^{j_2} \text{ y } V_{65}^{j_1} = 2^{\mu_3} V_{32}^{j_2}$$

Lo anterior se puede demostrar sin más que tomar la definición de los V_{kl}^{j}'s. Pero, además, es una condición suficiente, ya que por la forma que tienen los vectores V_{kl}^{j}'s (los vectores M_j^1's son independientes) se cumple una de las cuatro condiciones anteriores solo si al menos un elemento no nulo de la fila j_1 comparte columna con un elemento no nulo de la fila j_2. Esta comprobación no se puede hacer directamente en las ecuaciones de la clave pública, ya que no se puede determinar la correspondencia entre los exponentes de los monomios de cada polinomio de la clave pública y los H_j^{kl}. Para ello se tienen que encontrar, de una manera indirecta, a través de los V_{kl}^{j}. Fijada la fila j, si se considera todas las diferencias

$$H_j^{kl} - H_j^{k'l'}$$

nos damos cuenta de que las cantidades que nos interesa comparar, los V_{kl}^{j}, son justamente los $H_j^{kl} - H_j^{k'l'}$ cuyo valor se repite 3 veces. Si contamos salen 36 diferencias del tipo $H_j^{kl} - H_j^{k'l'}$, de las cuales 18 son de la forma V_{kl}^{j} (de hecho son 6 de la forma V_{kl}^{j} repetidas 3 veces módulo el signo). Los 18 restantes son diferencias del tipo

$$v_i - v_{i'} - v_k - v_{k'}$$

las cuales no interesan. Además la probabilidad de que se repitan es muy pequeña. Con esta observación se tiene para cada j el listado de los V_{kl}^{j} con los que se puede testar y determinar qué elementos no nulos comparten columna. Por tanto, con las ecuaciones de la clave pública se puede obtener qué elementos no nulos comparten columna y su cociente. Pero no se obtiene ninguna información sobre la columna a la que pertenecen ni su posición relativa en la fila, pero eso no es un problema.

Ahora se analizará la estructura del sistema. Una pregunta que normalmente no tendría sentido en sistemas de clave pública surge con este tipo de sistemas tan elaborados. ¿Qué suposiciones se pueden hacer sobre la clave privada? ¿Qué datos se pueden conocer a priori? Vamos a ver que se puede ir haciendo suposiciones sobre el sistema sin tener por que conocer nada de él.

Partimos de un sistema con clave pública $\{H_i : 1 \leq i \leq m\}$ y clave privada

$$\mathbb{S} = \{C, Y\},$$

donde

$$Y = \begin{cases} y_1 = x^{M_1^1} + x^{M_1^2} + x^{M_1^3} \\ \vdots \\ y_j = x^{M_j^1} + x^{M_j^2} + x^{M_j^3} \\ \vdots \\ y_n = x^{M_n^1} + x^{M_n^2} + x^{M_n^3} \\ y_{n+1} = x^{M_{n+1}} \\ \vdots \\ y_m = x^{M_m} \end{cases}.$$

Además, definimos

$$M = \begin{pmatrix} M_{n+1} \\ \vdots \\ M_m \end{pmatrix}.$$

Como ya se ha comentado anteriormente es suficiente considerar como clave privada $\mathbb{S}$, ya que con dicha información somos capaces de invertir el sistema. Ahora veamos qué suposiciones sobre la clave privada se pueden hacer. Para ello se buscarán operaciones que dejen invariante el sistema, i.e, se buscará un sistema equivalente al que se pueda llegar haciendo transformaciones sobre los elementos de la clave privada, aunque no se sepa calcular dichas operaciones. Este sistema equivalente ha de verificar:

1. Debe tener las mismas ecuaciones (pueden estar permutadas), para que la clave pública sea la misma.

2. El sistema debe tener la misma forma y propiedades, ya que, a priori, solo se sabe resolver sistemas de esa forma. Esto quiere decir que la matriz C' del nuevo sistema debe tener rango máximo y el sistema Y ha de verificar la 'estructura triangular'.

Por tanto, se estudiará qué tipo de operaciones dejan invariante un sistema inicial dado. De esta manera se puede, paso a paso, ir de nuestro sistema inicial (desconocido) a un

sistema que es equivalente y del cual se conoce su clave privada asociada, para finalmente trabajar sobre este nuevo sistema equivalente del cual se conoce ciertos datos.

Para ello veamos una serie de transformaciones que dejan invariante cualquier sistema de este tipo.

Observación 5.7. Permutación de las columnas de C

Comprobemos que es irrelevante la posición de las columnas de C (con algunas restricciones), lo que significa que un atacante puede elegir la posición de las columnas independientemente de la posición real.

Sea T una matriz de permutación de la forma

$$T = \begin{pmatrix} \ddots & & & & \\ & 0 & 1 & & \\ & 1 & 0 & & \\ & & & \ddots & \\ & & & & Id_{m-n} \end{pmatrix} \quad \text{o} \quad T = \begin{pmatrix} Id_n & & & & \\ & \ddots & & & \\ & & 0 & 1 & \\ & & 1 & 0 & \\ & & & & \ddots \end{pmatrix}$$

Si se considera el sistema $\mathbb{S}' = \{CT, T*Y\}$ se tiene que las ecuaciones resultantes son las mismas ya que, por la ecuación 5.1.2, se obtiene

$$P = C * Y = (CTT) * Y = (CT) * (T * Y)$$

Además, las propiedades que verificaba C quedan invariantes. Esto dice que se puede hacer hacer ciertas permutaciones en las coordenadas de y.

Observación 5.8. Reducción de la parte izquierda de C

Supongamos que sacamos factor común 2^{λ_j} a la j-ésima columna de la parte izquierda de C, donde 2^{λ_j} es el valor del primer elemento no nulo de dicha columna, el cual se sabe que existe porque C tiene rango máximo. Entonces, la columna j-ésima de C, c_j, se puede poner como:

$$c_j = c'_j 2^{\lambda_j} = 2^{\lambda_j} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 2^{\lambda_{lj}} \\ \vdots \\ 0 \end{pmatrix}$$

Esto quiere decir que C se puede escribir como

$$C = \begin{pmatrix} c'_1 & \dots & c'_n & c_{n+1} & \dots & c_m \end{pmatrix} \begin{pmatrix} 2^{\lambda_1} & & & & \\ & \ddots & & & \\ & & 2^{\lambda_n} & & \\ & & & Id_{m-n} \end{pmatrix}$$

donde en cada columna c'_j aparece un 1 como primer elemento no nulo. Sea

$$T = \begin{pmatrix} 2^{\lambda_1} & & & \\ & \ddots & & \\ & & 2^{\lambda_n} & \\ & & & Id_{m-n} \end{pmatrix},$$

y consideremos $\mathbb{S}' = \{CT^{-1}, TY\}$, por lo mismo que la observación anterior se tiene que las ecuaciones de ambos sistemas son las mismas ya que:

$$P = C * Y = (CT^{-1}T) * Y = (CT^{-1}) * (T * Y)$$

Por otra parte, por la forma que tiene T^{-1} y T, se tiene que $C' = CT^{-1}$ y $Y' = TY$ tienen las mismas propiedades que C e Y. Con esto se obtiene que el sistema

$$\mathbb{S}' = \{CT^{-1}, TY\}$$

es equivalente al de partida. Esto muestra que sin pérdida de generalidad se puede suponer que toda columna de la parte izquierda de C tiene un 1 como primer elemento no nulo.

Observación 5.9. Transformación de C_{22}.

Veamos que se tiene libertad de elección, siempre que se preserve el rango, en las entradas de la matriz C_{22}. Sea

$$T = \begin{pmatrix} Id_n & 0 \\ 0 & B \end{pmatrix}$$

con B de rango máximo. Si se considera el sistema $\mathbb{S}' = \{C', Y'\} = \{CT^{-1}, TY\}$, se ve que las ecuaciones del sistema son las mismas (análogo al caso anterior, solo hay que escribir las ecuaciones). Además, como

$$T^{-1} = \begin{pmatrix} Id_n & 0 \\ 0 & B^{-1} \end{pmatrix}$$

se tiene que la forma de la matriz de C' es la misma que la de C ya que la parte izquierda de ambas es la misma y la parte derecha preserva el rango máximo.

El sistema Y sigue verificando la estructura triangular, ya que $M_j^k = M_j'^k \ \forall j = 1, \dots n$ y $\forall k = 1, 2, 3$ y como B tiene rango máximo también se verifica que:

$$L\left[\{M_{n+1}, \dots, M_m\}\right] = L\left[\left\{M'_{n+1}, \dots, M'_m\right\}\right]$$

con lo que solamente se está haciendo un cambio de base.

Observación 5.10. Transformación de C_{12}.

Veamos, como en la observación anterior, que se tiene una cierta libertad de elección en las entradas de la matriz C_{12}. Sea

$$T = \begin{pmatrix} Id_n & A \\ 0 & Id_{m-n} \end{pmatrix}$$

y se considerará el nuevo sistema

$$\mathbb{S}' = \left\{C', Y'\right\} = \left\{CT^{-1}, TY\right\}$$

En este caso el cambio es

$$M_j'^k = M_j^k + A(j)M$$

$\forall j = 1, \dots n$ y $\forall k = 1, 2, 3$, lo cual quiere decir que la 'estructura triangular' se sigue verificando ya que

$$L\left[\{M_{n+1}, \dots, M_m\}\right] = L\left[\left\{M'_{n+1}, \dots, M'_m\right\}\right]$$

Además, se puede ver de manera similar a las observaciones, que la forma de C' se sigue manteniendo ya que

$$T^{-1} = \begin{pmatrix} Id_n & -A \\ 0 & Id_{m-n} \end{pmatrix}$$

Supongamos que se considera (siempre y cuando C_{11} tenga inversa, lo cual siempre se puede suponer haciendo un intercambio de filas, ya que eso solo repercutiría en un intercambio de posición en las ecuaciones de la clave pública)

$$T = \begin{pmatrix} Id_n & -C_{11}^{-1}C_{12} \\ 0 & Id_{m-n} \end{pmatrix}$$

entonces se tiene que

$$C' = CT = \begin{pmatrix} C_{11} & 0 \\ C_{21} & \star \end{pmatrix}$$

Esto quiere decir que hay otro sistema en el que $C'_{12} = 0$ y preserva la parte izquierda de C. Además, un hecho muy importante, es que $M = M'$.

Observación 5.11. Reducción de C_{22}.

Veamos, sin pérdida de generalidad, que se puede suponer que $C_{22} = Id_{m-n}$. Esto es un caso particular de la observación 5.9.

Sea

$$T = \begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix}$$

y considérese el nuevo sistema

$$\mathbb{S}' = \left\{C', Y'\right\} = \left\{CT^{-1}, TY\right\}$$

En este caso, el cambio que se hace es

$$\begin{cases} y_{n+1} = x^{C_{22}(1)M} \\ \vdots \\ y_m = x^{C_{22}(m-n)M} \end{cases}$$

Como la matriz C_{22} tiene rango máximo la estructura triangular del sistema no varía, por lo que se obtiene un sistema equivalente.

Estas observaciones son la clave principal del ataque.

5.4.3. Análisis

A continuación se describirá los pasos del ataque utilizando las transformaciones mostradas en las observaciones anteriores. Se ha visto que se puede calcular qué elementos no nulos de la parte izquierda de C comparten columna así como los cocientes entre ellos, pero no en qué columna están ni cual es su valor. Por la observación 5.7 se puede elegir la posición de cada columna. De esta manera y con lo anterior se obtienen las posiciones de los elementos no nulos y los cocientes entre los elementos de una misma columna. Con esto se llega a conocer cada columna módulo una constante. Si al primer elemento no nulo de la columna

c_j lo llamamos r_j se tiene que la columna c_j es de la forma:

$$\begin{pmatrix} 0 \\ \vdots \\ r_j \\ \vdots \\ 2^{s_j^1} r_j \\ \vdots \end{pmatrix}$$

Por la observación 5.8 se puede suponer, sin pérdida de generalidad, que sacamos factor común r_j a la columna j-ésima para cada j y así se obtiene un 1 como primer elemento no nulo. Como se sabe el cociente entre los elementos de una misma columna ($2^{s_j^l}$'s) se pueden hallar todos los elementos no nulos de una misma columna y, por tanto, saber el valor de todas las entradas de la parte izquierda de C.

Con todo lo anterior llegamos a obtener las entradas de las matrices C_{11} y C_{21}.

Por la observación 5.10 se puede suponer, sin pérdida de generalidad, que nuestro sistema verifica que $C_{12} = 0$ y, además, la parte izquierda de C es la misma.

Considérese el siguiente sistema

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} M^1 = H^{11}$$

donde C_{11}, C_{21} y C_{12} son conocidos. Ahora queremos aplicar Gauss a la parte izquierda de la matriz C para obtener

$$\begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix} M^1 = \bar{H}^{11}$$

Pero, ¿como se puede hacer Gauss al sistema? Considérese G la matriz de manera que:

$$G \begin{pmatrix} C_{11} & 0 \\ C_{21} & C_{22} \end{pmatrix} \equiv \begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix} \quad \text{mód } q-1$$

Como comentamos antes se tiene que

$$B * (A * x) = (BA) * x$$

con lo cual, si $z = C * Y$, se obtiene (siempre y cuando z tenga todas su coordenadas no

nulas, para que los exponentes estén en $\mathbb{Z}_{q-1}$)

$$G{*}z = G * (C * Y) = (GC) * Y = \begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix} * Y$$

Esto quiere decir que si se hace la transformación $G*$ a la clave pública se obtiene

$$G * z = \begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix} * Y = \begin{cases} y_1 = x^{M_1^1} + x^{M_1^2} + x^{M_1^3} \\ \vdots \\ y_j = x^{M_j^1} + x^{M_j^2} + x^{M_j^3} \\ \vdots \\ y_n = x^{M_n^1} + x^{M_n^2} + x^{M_n^3} \\ y_{n+1} = x^{C_{22}(1)M} \\ \vdots \\ y_m = x^{C_{22}(m-n)M} \end{cases}$$

Se tiene que C_{22} tiene rango máximo, por tanto por la observación 5.11 si se considera

$$T = \begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix}$$

el sistema $\mathbb{S}' = \left\{CT^{-1}, TY\right\}$ es equivalente al nuestro. Esto quiere decir que podemos quedarnos con el sistema $\mathbb{S}'$ sin perder la información que ya se tiene. De esta forma se ha llegado a un sistema donde

$$C' = CT^{-1} = \begin{pmatrix} C_{11} & 0 \\ C_{21} & Id_{m-n} \end{pmatrix}$$

con C_{11} y C_{21} conocidos.

Aunque este planteamiento teórico es correcto y funciona para q y m pequeños, cuando nosotros calculamos $G{*}z$ aparece una cantidad de monomios muy grande con la cual no se puede trabajar y, por tanto, no se puede hacer simplificaciones. Es decir, cuando se calcula $G{*}z$ se está obteniendo

$$G(i_0){*}z = \left(x^{M_{i_0}^1} + x^{M_{i_0}^2} + x^{M_{i_o}^3}\right) \prod_{1 \leq i \leq m} \left(x^{M_i^1} + x^{M_i^2} + x^{M_i^3}\right)^{\lambda_i(q-1)} \tag{5.4.1}$$

desarrollado, donde

$$\prod_{1\leq i\leq m}\left(x^{M_i^1}+x^{M_i^2}+x^{M_i^3}\right)^{\lambda_i(q-1)}=1.$$

Por tanto, una vez calculado se tiene que sacar factor común y quedarnos solamente con $x^{M_{i0}^1}+x^{M_{i0}^2}+x^{M_{io}^3}$. Pero esto no es posible, ya que al desarrollar dicha expresión se obtiene una cantidad de monomios que crece de manera exponencial en p, por lo que no es posible trabajar con dicha expresión. En cambio, lo que se obtiene al calcular sin desarrollar $G*z=(f_1,\ldots,f_m)$ son funciones de la forma

$$f_i=\prod_{1\leq j\leq m}\left(\sum_{k,l=1}^{3}x^{H_j^{kl}}\right)^{G(i,j)}$$

de manera que

$$f_{i|W}=y_{i|W}.$$

Además, se puede evaluar las f_i's sobre puntos de manera eficiente, gracias a ello se puede obtener más información del sistema. Para $j=n+1,\ldots,m$ se tiene que

$$y_j=x^{M_j},$$

por lo que evaluando la función f_j en puntos de la forma $(1,\ldots,1,g,1,\ldots,1)$ donde g ocupa la posición i-ésima y es un generador del grupo multiplicativo $\mathbb{F}_q^*$ se puede obtener el exponente $M_j(i)$. Para obtener el exponente $M_j(i)$ se tiene que resolver la ecuación

$$g^{M_j(i)}=g'$$

que es hallar un logaritmo discreto. En este caso particular la complejidad es $O\left(2^{p/2}\right)=O\left(\sqrt{q}\right)$, suponiendo que el $q-1$ es primo ya que en otro caso el exponente se puede hallar con una complejidad menor tomando distintos elementos g que tengan un orden divisor del orden del grupo. Procediendo de la misma manera para el resto de exponentes y monomios se puede obtener el conjunto

$$\{M_j : n+1\leq j\leq m\}$$

Con esta información y con las suposiciones que se han hecho previamente se puede encontrar la información que falta para resolver el sistema. Definimos

$$U_0=L\left[\{M_j : n+1\leq j\leq m\}\right]$$

Considérese ahora los vectores de la clave pública H_j^{kl} y todas las diferencias

$$H_j^{kl} - H_j^{k'l'} \quad \forall k, l, k', l' = 1, 2, 3. \tag{5.4.2}$$

Se sabe que en estas diferencias se encuentran los vectores V_{kl}^j mencionados anteriormente. La estructura triangular asegura que existe al menos un s_0 tal que

$$M_{s_0}^2, M_{s_0}^3 \in U_0.$$

Por otra parte, como ninguna columna de C es nula se sabe que existe un j_0 de manera que

$$M_{s_0}^3 - M_{s_0}^2 = V_{32}^{j_0} \text{ o } M_{s_0}^3 - M_{s_0}^2 = V_{65}^{j_0}$$

Esto quiere decir que, si para cada j se hacen todas las diferencias de la forma 5.4.2, se puede encontrar el vector $\pm\left(M_{s_0}^3 - M_{s_0}^2\right)$ comprobando si las diferencias pertenecen o no al espacio vectorial U_0. Para un j dado la manera de proceder sería la siguiente:

- Calculamos las 36 diferencias posibles, $H_j^{kl} - H_j^{k'l'}$, de los 9 exponentes H_j^{kl}.
- Con cada diferencia se comprueba si pertenece o no al espacio U_0. Cuando una de ellas pertenezca se tiene que otras dos también pertenecen a U_0. De hecho estas tres diferencias son iguales salvo el signo. En ese caso $j = j_0$.
- Se sabe que los vectores que se han encontrado son $\pm\left(M_{s_0}^3 - M_{s_0}^2\right)$. Se puede suponer que el signo es positivo, ya que como luego se verá esto solo afecta al hecho de que se saque factor común $x^{M_{s_0}^2}$, pero en caso de tener el vector signo negativo lo que estaríamos haciendo sería sacar factor común $x^{M_{s_0}^3}$. Sin pérdida de generalidad, supongamos que $M_{s_0}^3 - M_{s_0}^2$ es de la forma $V_{32}^{j_0}$ (análogo para el caso $V_{65}^{j_0}$). En ese caso se tiene que los 3 vectores que se ha encontrado son:

  $$\begin{aligned} M_{s_0}^3 - M_{s_0}^2 &= H_{j_0}^{31} - H_{j_0}^{21} \\ M_{s_0}^3 - M_{s_0}^2 &= H_{j_0}^{32} - H_{j_0}^{22} \\ M_{s_0}^3 - M_{s_0}^2 &= H_{j_0}^{33} - H_{j_0}^{23} \end{aligned}$$

  Somos capaces de hallar las diferencias entre los vectores, pero no se puede hallar los vectores en sí, ya que el rango de las diferencias que si podemos hallar es menor que el rango de vectores. Esto, en particular , dice que se han encontrado también los conjuntos $H_{j_0}^{1l}$, $H_{j_0}^{2l}$ y $H_{j_0}^{3l}$.
- Ahora se calculan todas las diferencias entre los elementos de $H_{j_0}^{1l}$ y $H_{j_0}^{2l}$. En las 9

diferencias se obtiene un vector que se repite 3 veces, ese vector es el $v_1 - v_2 = M^1_{s_0} - M^2_{s_0}$. Ningún otro vector se repite tres veces, solo hay que mirar los posibles casos. En este caso se obtiene con signo positivo el vector $M^1_{s_0} - M^2_{s_0}$.

- Finalmente se calcula

$$U_1 = L\left[M^1_{s_0} - M^2_{s_0}\right] \oplus U_0 = L\left[M^1_{s_0}\right] \oplus U_0$$

Obteniendo así $M^3_{s_0} - M^2_{s_0}$, $M^1_{s_0} - M^2_{s_0}$ y U_1.

Llegados a este punto la estructura triangular asegura que existe al menos un $s_1 \notin \{s_0\}$ tal que:

$$M^2_{s_1}, M^3_{s_1} \in U_1.$$

Procediendo de manera recursiva hasta obtener los conjuntos

$$D_1 = \left\{M^1_j - M^2_j : j = 1, \ldots, n\right\} \text{ y } D_2 = \left\{M^3_j - M^2_j : j = 1, \ldots, n\right\}.$$

De esta manera se obtiene los y_j salvo multiplicación por un monomio ya que:

$$y_j = \underbrace{x^{M^2_j}}_{\text{Desconocido}} \left(\underbrace{x^{M^1_j - M^2_j} + x^{M^3_j - M^2_j} + 1}_{\text{Conocido}}\right)$$

En este caso no se puede suponer nada sobre M^2_j, ya que cualquier suposición haría cambiar las ecuaciones del sistema, por lo que se debe hallar su valor de una manera indirecta. Antes supusimos que el vector que habíamos encontrado era $+(M^3_j - M^2_j)$, en caso de no ser así entonces habríamos encontrado $+(M^2_j - M^3_j)$ lo cual hace que la parte no conocida de y_j sea $x^{M^3_j}$ y no $x^{M^2_j}$, pero ese hecho no cambia nada la argumentación. Finalmente, se utiliza las f_j halladas previamente para hallar los $x^{M^2_j}$ sin más que considerar la siguiente relación

$$x^{M^2_j}(p) = \frac{f_j(p)}{g_j(p)}$$

siendo

$$g_j = x^{M^1_j - M^2_j} + x^{M^3_j - M^2_j} + 1$$

Así, se pueden obtener las ecuaciones de y y, por tanto aplicando la propiedad triangular y haciendo los pasos mostrados en el proceso de desencriptado, podremos obtener el valor de x. Todo este proceso muestra que únicamente sabiendo la clave pública y un valor cifrado podemos recuperar el texto original en tiempo polinomial, aunque no calculamos

explícitamente la inversa del sistema. Con estos argumentos mostramos que el sistema ME no es seguro.

5.4.4. Observaciones

El método es un tanto complicado, pero se explicará cómo otro tipo de posibles métodos más sencillos fallan en la práctica.

Observación 5.12. Otra manera de proceder a priori utilizando todo lo anterior, es la siguiente. Con las consideraciones previas se sabe que z_i es de la forma

$$z_i = \left(x^{M_{j_1}^1} + x^{M_{j_1}^2} + x^{M_{j_1}^3}\right)^{a_i} \left(x^{M_{j_2}^1} + x^{M_{j_2}^2} + x^{M_{j_2}^3}\right)^{b_i} x^{N_i} \tag{5.4.3}$$

siendo $N_i = 0$ si $1 \leq i \leq n$. Lo primero que se nos ocurre sería factorizar la expresión 5.4.3, obteniendo así:

$$\left(x^{\bar{M}_{j_1}^1} + x^{\bar{M}_{j_1}^2} + x^{\bar{M}_{j_1}^3}\right)^{a_i} \left(x^{\bar{M}_{j_2}^1} + x^{\bar{M}_{j_2}^2} + x^{\bar{M}_{j_2}^3}\right)^{b_i} x^{\bar{N}_i} \tag{5.4.4}$$

De esta manera se puede obtener, con una probabilidad muy alta, los a_i's y b_i's. Las expresiones 5.4.3 y 5.4.4 en principio no son iguales ya que si sacamos factor común se obtiene:

$$\left(x^{M_{j_1}^1} + x^{M_{j_1}^2} + x^{M_{j_1}^3}\right)^{a_i} = x^{a_i \tilde{N}_{j_1}} \left(x^{\bar{M}_{j_1}^1} + x^{\bar{M}_{j_1}^2} + x^{\bar{M}_{j_1}^3}\right)^{a_i}$$

análogo con la otra expresión. Con lo que

$$\bar{N}_i = a_i \tilde{N}_{j_1} + b_i \tilde{N}_{j_2} + N_i \tag{5.4.5}$$

Supongamos que $1 \leq i \leq n$, en ese caso $N_i = 0$. Por tanto, para hallar el valor de los M_j^k's utilizando los valores conocidos $\bar{M}_j^k$'s y $\bar{N}_j$'s se tiene que calcular los valores $\tilde{N}_j$'s. Por la ecuación 5.4.5 se tiene que:

$$C_{11} \begin{pmatrix} \tilde{N}_1 \\ \vdots \\ \tilde{N}_n \end{pmatrix} = \begin{pmatrix} \bar{N}_1 \\ \vdots \\ \bar{N}_n \end{pmatrix}$$

Como C_{11} es invertible, se puede calcular los N_j's. Una vez calculados se pasa a hallar el valor de N_i para $n+1 \leq i \leq m$. Pero de nuevo, este planteamiento teórico no se puede llevar a la práctica ya que en la clave pública los exponentes están módulo $q-1$, por lo que no se puede factorizar 5.4.3 para sacar la expresión 5.4.4. Si los exponentes no fuesen módulo $q-1$, este método rompería el sistema ya que se puede obtener la expresión 5.4.4

para los valores de q y m pensados para el sistema.

Observación 5.13. Otra idea que se puede aplicar y que tampoco funciona es considerar los sistemas lineales

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} M^1 = H^{11}$$

para hacer Gauss y obtener

$$\begin{pmatrix} Id_n & 0 \\ 0 & C_{22} \end{pmatrix} M^1 = \bar{H}^{11}$$

Con este método se pueden hallar los vectores de M^1 y análogamente con M_2 y M_3. Pero este método no se puede aplicar, ya que los exponentes de la clave pública no vienen ordenados y, por tanto, no se tiene ninguna manera, a priori, de determinar la matriz H^{11}.

Observación 5.14. Aunque se pongan coeficientes en los monomios de la variable y, el sistema sigue estando roto, sin más que hacer unas leves modificaciones en el análisis anterior.

Observación 5.15. Todas las modificaciones que se han ido haciendo al sistema para que sea resistente a este ataque no han surtido efecto, ya que siempre se ha podido obtener información total o parcial de C y con ello se ha podido ir hacia atrás. Con lo que concluimos que este tipo de sistemas no son seguros.

Capítulo 6

Otros sistemas

Ahora presentaremos dos sistemas más en los que hemos trabajado, siguiendo con la idea de aplicaciones exponenciales o aplicaciones triangulares. Para el primero he conseguido desarrollar un método que muestra que no es seguro, pero el segundo aún no he podido romperlo.

6.1. Sistema HFE + exponencial

Uno de los sistemas más estudiados y con un gran potencial en su momento fue el HFE, desarrollado por Patarin. El HFE surge como una generalización del criptosistema Matsumoto-Imai. La idea en la que se basa el HFE es utilizar la estructura de $\mathbb{K}$-espacio vectorial de dimensión n que tiene $\mathbb{E}$ para ocultar la estructura de $\mathbb{E}$.

Como ya se vio en su momento, el HFE no es seguro ya que las bases de Groebner pueden encontrar las soluciones del sistema asociado a la clave pública. Esto es así porque el índice de regularidad del sistema se mantiene constante si limitamos el grado de la ecuación univariada del HFE. Si se quiere modificar el HFE para que sirva como sistema criptográfico se tiene que hacer las modificaciones oportunas para que el índice de regularidad aumente a medida que aumentamos el número de variables. La idea que se propone es combinar el HFE con la aplicación exponencial $*$, lo cual produce que los polinomios del sistema tengan un grado elevado y, por tanto, el índice de regularidad se dispare.

Sea $x = A * u$, donde A es una matriz de rango máximo con entradas en $\mathbb{Z}_{q-1}$. Como pasaba con el otro sistema, para poder utilizar la aplicación exponencial es necesario que ninguna de las entradas de la variable x sea nula. La idea es sustituir el valor de la x en

las ecuaciones de la clave pública del HFE por $A * u$. En este caso, a diferencia del HFE normal, las bases de Groebner no consiguen obtener las soluciones del sistema, ya que en este caso se puede comprobar que el índice de regularidad es muy elevado.

Análisis

Nosotros partimos de un sistema de ecuaciones de la forma

$$f_k = \sum a_{ij} x_i x_j$$

y al hacer el cambio $x = A * u$ se obtiene un sistema de la forma

$$f_k = \sum a_{ij} u^{A_i} u^{A_j}.$$

A este nuevo sistema no se le puede aplicar las bases de Groebner como hacíamos con el HFE ya que ahora el índice de regularidad ha aumentado considerablemente. Pero como se verá, esto no es suficiente, ya que se puede obtener las filas de la matriz A y con ello recuperar el sistema HFE inicial como se verá al final. Las filas de la matriz A están ocultas pero no así el conjunto

$$\mathcal{U} = \{A_i + A_j : 1 \leq i, j \leq n\}$$

que se puede obtener directamente de la clave pública. Lo primero que se hará, será encontrar las filas de la matriz A, para ello se debe distinguir entre si $q-1$ es par o impar.

- Si q es de la forma 2^p entonces la matriz A tiene sus entradas en $\mathbb{Z}_{q-1}$ con $q-1$ impar. Por tanto, se puede dividir los elementos de $\mathcal{U}$ entre 2, ya que en ese caso el 2 tiene inverso. La manera de proceder sería la siguiente:

  1. Calculamos el conjunto

     $$\mathcal{U}' = \left\{(A_i + A_j)\, 2^{-1} : 1 \leq i, j \leq n\right\}$$

     Se tiene que $A_i \in \mathcal{U}'$ para todo i.
  2. Para cada $A_i + A_j$ en $\mathcal{U}$ se buscan las posibles parejas a y b en $\mathcal{U}'$ tal que $a + b = A_i + A_j$.
  3. Ahora se dan 2 casos:

- Si $i = j$: Como los A_i son linealmente independientes no existe una combinación lineal de la forma

$$2A_i = (A_{i_1} + A_{j_1})\,2^{-1} + (A_{i_2} + A_{j_2})\,2^{-1}$$

ya que sino llegaríamos a una contradicción con el rango de A, a no ser que todos los subíndices coincidan. Por tanto, $2A_i$ solo se puede poner como suma de $A_i + A_i$ con A_i en $\mathcal{U}'$. De esta manera al buscar los a y b que cumplan la condición anterior solo se encontrará un a tal que $2A_i = a + a$. De esta manera se ha encontrado $a = A_i$, que se guardará en la lista L.
- Si $i \neq j$: Por el mismo argumento anterior $A_i + A_j$ solo se puede expresar como suma de elementos de $\mathcal{U}'$ como $A_i + A_j$ o $(A_i + A_j)\,2^{-1} + (A_i + A_j)\,2^{-1}$. Por tanto, al buscar los elementos a y b que cumplan la condición anterior solo se encontrará un par a y b distintos tal que $A_i + A_j = a + b$. De esta manera se ha encontrado $a = A_i$ y $b = A_j$, que se guardarán en la lista L.

4. Una vez la lista L tenga n elementos independientes se habrán encontrado las filas de la matriz A.

Por tanto, se ha obtenido PA que es la matriz A salvo una permutación de sus filas. Calculando la inversa de esta matriz se obtiene

$$(PA)^{-1} = A^{-1}P$$

Como se verá en el siguiente apartado con esto se puede de resolver el sistema.

- Si q no es de la forma 2^p entonces la matriz A tiene sus entradas en $\mathbb{Z}_{q-1}$ con $q-1$ par, en este caso no podemos proceder de la misma manera que antes. Lo primero que se utiliza es el hecho de que los elementos de $\mathcal{U}$ que son de la forma $2A_i$ tienen todas sus entradas pares. Lo cual en el caso anterior pasaba siempre con cualquier elemento ya que el 2 era una unidad, pero no en este caso. Como los elementos de la forma $2A_i$ tienen todas sus entradas pares, se les puede identificar en el conjunto $\mathcal{U}$ de manera rápida. Podría darse el caso de que un elemento de la forma $A_i + A_j$ con $i \neq j$ tuviese todas sus entradas pares, pero como los elementos A_i y A_j son aleatorios esto solo se daría con una probabilidad de $1/(q-1)^n$. Como el número de elementos en $\mathcal{U}$ que no son de la forma $2A_i$ es $n(n-1)$, es muy poco probable que se encuentren falsos positivos. Por tanto, se puede hallar el conjunto

$$\mathcal{W} = \{2A_i : 1 \leq i \leq n\}$$

Llegados a este punto supongamos que se tienen n elementos linealmente independientes de $\mathcal{U} \setminus \mathcal{W}$, llamémoslos $v_1, \ldots, v_n$. Se sabe que $v_k = A(e_{i_k} + e_{j_k})$, por tanto, existe una transformación lineal que lleva la matriz V, cuyas columnas son los vectores v_i, a una matriz B que solo tiene ceros y unos, y en cada fila solo tiene dos elementos no nulos. Por ejemplo, la transformación lineal asociada a la matriz A^{-1}, en este caso la fila k-ésima de la matriz B es justamente $e_{i_k} + e_{j_k}$. Lo que se hará será algo similar, es decir, buscar una matriz C cuya transformación lineal asociado mande cada elemento de $\mathcal{U} \setminus \mathcal{W}$ a un elemento de la forma $(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0)$. Con ello conseguimos una matriz C de manera que

$$C * u^{A_i + A_j} = x_{k_i} x_{k_j}$$

Dicha matriz no tiene por que ser A^{-1}, lo que se hará es buscar a una matriz que se comporte como A^{-1} salvo una permutación. Por otra parte, no vale cualquier matriz, ya que entonces al hacer la sustitución $u = C * z$ las ecuaciones no quedan todas cuadráticas si la matriz no está bien elegida. La manera de proceder para encontrar la matriz C sería la siguiente:

1. Ordenamos el conjunto $\mathcal{W}$, obteniendo $\mathcal{W} = \{w_l : 1 \leq l \leq n\}$.
2. Se sabe que $v_k = A_{i_k} + A_{j_k}$, por tanto, existen $w_{l_{k_1}}$ y $w_{l_{k_2}}$ de manera que $2v_k = w_{l_{k_1}} + w_{l_{k_2}}$. Como $\mathcal{W}$ no es base podría darse el caso en el que existan más parejas de $w's$ que cumplan la condición anterior. Solo hay una pareja que vale, por lo que se tiene que probar con todas y desechar las que no valgan. Pero como antes, la probabilidad de obtener un falso positivo es muy pequeña. Esto es así porque si se diese el caso se tendría

   $$w_l + w_k - w_{l'} - w_{k'} = 2 * (v_h + v_t - v_{h'} - v_{t'}) = 0$$

   Y para que se de dicha igualdad todas las entradas del vector $v_h + v_t - v_{h'} - v_{t'}$ deberían ser iguales a $(q-1)/2$ o 0.
3. Se genera la matriz B, donde la fila k-ésima es el vector $e_{l_{k_1}} + e_{l_{k_2}}$ ($v_k = w_{l_{k_1}} + w_{l_{k_2}}$).
4. Llegados a este punto se tiene que la matriz C es

   $$C = V^{-1} B$$

Veamos porqué. Sea V la matriz conocida cuyas columnas son los vectores v_i y sea

B' la matriz desconocida cuyas filas son los elementos $v_i A^{-1}$, es decir, $B' = VA^{-1}$. Si se supiese que forma tiene B' entonces se puede encontrar A^{-1}, ya que

$$A^{-1} = V^{-1}B'$$

Esto no es posible, ya que no se puede obtener exactamente la matriz B'. Lo que se obtiene con este procedimiento es una matriz muy similar

$$B = B'P$$

donde P es una matriz de permutación. Esa matriz de permutación aparece porque al ordenar el conjunto $\mathcal{W}$ se está introduciendo una permutación σ, de manera que:

$$2A_i = w_{\sigma(i)}$$

Por lo que P es la matriz asociada a la permutación σ. Con esto se logra obtener un sistema que es igual al inicial salvo una permutación en sus variables, lo cual no es un problema. Gracias a B se puede encontrar A^{-1} (salvo una permutación) sabiendo que

$$V^{-1}B = V^{-1}B'P = A^{-1}P$$

Llegados a este punto sustituimos

$$u = C * z = \left(V^{-1}B\right) * z = \left(A^{-1}P\right) * z$$

Como $x = A * u$ se tiene que:

$$x = A * u = A * (A^{-1}P * z) = (AA^{-1}P) * z = P * z$$

Se sabe que $A_i + A_j$ era el exponentes de $x_i x_j$ tras aplicar $*A$. Lo que se está haciendo es el procedimiento inverso salvo una permutación de los subíndices.

$$e_i + e_j \xrightarrow{A} A_i + A_j \xrightarrow{V^{-1}B} e_{\sigma(i)} + e_{\sigma(j)}$$

Que en las variables sería:

$$x_i x_j \xrightarrow{A*} u^{A_i + A_j} \xrightarrow{V^{-1}B*} z_{\sigma(i)} z_{\sigma(j)}$$

De esta forma se obtiene un sistema HFE, el cual se puede resolver mediante el

algoritmo F4, por lo que el sistema queda roto.

6.2. Aplicaciones triangulares

Ahora vamos a presentar un nuevo sistema MPKC, al cual llamaremos GTS (Generic Triangular System). Basado en aplicaciones triangulares desarrollado íntegramente por mí y al cual no le hemos encontrado ninguna vulnerabilidad.

Veamos ahora unas propiedades básicas de los automorfismos triangulares. Este tipo de aplicaciones surgen en la geometría algebraica. Se sabe que el grupo de automorfismos de $\mathbb{K}^2$ está generado por los automorfismos triangulares, pero para dimensiones mayores no es cierto (Automorfismo de Nagata). En cualquier caso, trabajamos con un subgrupo del grupo de automorfismos. Denotamos por $\mathbb{K}$ el cuerpo finito de 2^m elementos. Sea, por tanto, $\mathbb{K}^n$ el espacio afín de dimensión n sobre $\mathbb{K}$, y $\mathbb{K}[x_1, \ldots, x_n]$ el anillo de polinomios sobre $\mathbb{K}$.

Definición 6.1. Una aplicación $\phi : \mathbb{K}^n \to \mathbb{K}^n$ es un automorfismo triangular, si verifica una de las siguientes condiciones:

1. La aplicación ϕ es una transformación afín invertible.
2. Haciendo una reordenación de las variables x_i, si fuese necesario, ϕ tiene la forma:

$$\begin{array}{ccc} \phi : \mathbb{K}^n & \longrightarrow & \mathbb{K}^n \\ x_1 & & y_1 = x_1 \\ x_2 & & y_2 = x_2 + h_2(x_1) \\ \vdots & \longrightarrow & \vdots \\ x_{n-1} & & y_{n-1} = x_{n-1} + h_{n-1}(x_1, \ldots x_{n-2}) \\ x_n & & y_n = x_n + h_n(x_1, \ldots, x_{n-1}) \end{array}$$

donde los h_i son polinomios.

Observación 6.2. Si $\phi : \mathbb{K}^n \to \mathbb{K}^n$ es un automorfismo triangular, entonces ϕ es invertible, y la preimagen de un punto cualquiera puede computarse en tiempo polinomial.

Demostración. Si ϕ cumple la primera condición, se tiene que es invertible por definición. Veámoslo en el caso en el que ϕ cumple la segunda condición. Si denotamos $\phi^{-1}(y) = (\phi_1^{-1}(y), \ldots, \phi_n^{-1}(y))$ entonces veamos por inducción que, para todo i existe ϕ_i^{-1} tal que $\phi_i^{-1}(y) = x_i$.

- Caso $i = 1$: definimos $\phi_1^{-1}(y) = y_1$.
- Supongamos ciertos los casos en los que $i < k$.
- Caso $i = k$: definimos $\phi_k^{-1}(y) = y_k - h_k(\phi_1^{-1}(y), \ldots, \phi_{k-1}^{-1}(y))$ y por hipótesis de inducción se tiene que $\phi_k^{-1}(y) = y_k - h_k(x_1, \ldots, x_{k-1}) = x_k$.

Esto también muestra que la preimagen de un punto se puede calcular de manera eficiente. □

Con esta observación, ahora tiene sentido hacer la siguiente definición.

Definición 6.3. El grupo generado por todos los automorfismos triangulares, se llama grupo de automorfismos tame, donde la operación del grupo es la composición de aplicaciones.

Observación 6.4. La observación 6.2 dice que la aplicación ϕ tiene inversa, además dice cómo calcularla. Por ejemplo, en el caso de 4 variables se tiene que:

$$\begin{cases} x_1 = y_1 \\ x_2 = y_2 - h_2(y_1) \\ x_3 = y_3 - h_3(y_1, y_2 - h_2(y_1)) \\ x_4 = y_4 - h_4(y_1, y_2 - h_2(y_1), y_3 - h_3(y_1, y_2 - h_2(y_1))) \end{cases}$$

Como se puede apreciar, si el $\deg(h_i) = 2$ para todo i y los h_i son aleatorios, entonces el $\deg(x_j)$, como polinomio en las y_i, crece de manera muy rápida. De hecho, se pueden escoger los h_i de grado 2 de manera que $\deg(x_j) = O(2^{2^j})$. Análogamente, se pueden escoger fácilmente los h_i de grado dos, tal que el número de términos en x_j, como polinomio en las y_i, sea $O(2^{2^j})$. Por tanto, tomando un n suficientemente grande, es muy difícil computacionalmente calcular explícitamente ϕ^{-1}. Sin embargo, dado un punto $(y_1, \ldots, y_n)$, se puede calcular fácilmente $\phi^{-1}(y)$. Por ejemplo, si los h_i son cuadráticos y con m términos, se podrá calcular $\phi^{-1}(y)$ en $O(n^3m^2)$ operaciones, haciendo el proceso inductivo de la demostración de la observación 6.2.

La idea del criptosistema es componer varias aplicaciones triangulares. Con esto se puede generar un sistema casi aleatorio, de hecho en las pruebas realizadas la complejidad del F4 sobre este sistema ha sido la misma que para sistemas aleatorios. Evidentemente el número de vueltas (composiciones) o número de aplicaciones triangulares que vamos a componer

es limitado, ya que el número de monomios crece muy rápidamente. La idea es trabajar con aplicaciones triangulares de la forma

$$I_i = I_i(j_1, j_2) = \begin{cases} y_1 = x_1 \\ \vdots \\ y_i = x_i + x_{j_1}^{2^{\lambda_1}} x_{j_2}^{2^{\lambda_2}} \\ \vdots \\ y_n = x_n \end{cases}$$

Ya que así se pueden dar muchas vueltas y mezclar bien el sistema. Los λ_i son aleatorios mód $q-1$ para así esconder una posible estructura en los exponentes de nuestros monomios. Utilizamos este tipo de aplicaciones y no aplicaciones triangulares genéricas ya que si no, solo se podría dar 2 o 3 vueltas lo cual sería un problema para la seguridad, además estas I_i son más generales. Trabajamos en cuerpos grandes, como por ejemplo $q = 2^{10}$, para que el número de incógnitas sea pequeño y el número de monomios no se dispare, ya que el controlar el número de monomios es la tarea más importante para desarrollar el sistema.

Por otra parte, que el número de variables sea pequeño no influye mucho en la complejidad del F4 sobre nuestro sistema, ya que al trabajar en un cuerpo grande y no poner restricción a los exponentes estos serán grandes. Por lo que el índice de regularidad es alto, y, por tanto, también el tiempo de ejecución del F4 sobre nuestro sistema.

La forma de nuestro sistema $F : \mathbb{K}^n \to \mathbb{K}^{2n}$ es por tanto:

$$F = S \circ \begin{pmatrix} I_{i_k} \circ \ldots \circ I_{i_1} \\ R \end{pmatrix}$$

donde S es una aplicación lineal invertible y R es un sistema aleatorio de n ecuaciones cuyos monomios son los mismos que los que aparecen en $I_{i_k} \circ \ldots \circ I_{i_1}$. Esto es útil para esconder la posible estructura que tuvieran los monomios de nuestro sistema. Con ello conseguimos que no haya monomios más relevantes que otros con los que poder sacar información de los coeficientes de S, ya que si se tuviesen varios grupos de monomios donde cada grupo aparece solo en una ecuación entonces se podría sacar información de las columnas de S. Al aplicar el F4 a nuestro sistema nos damos cuenta de que la complejidad es muy similar si se mete o no el sistema R, por lo que no supone una desventaja frente al $F4$.

Para generar el sistema, primero se tendrá que decidir una cota inferior y superior para el número de monomios en cada ecuación. Estas cotas están totalmente relacionadas con el tamaño de la clave del sistema así como con la seguridad del mismo. Para que el número

de monomios no se dispare hay que tener en cuenta varias consideraciones:

1. Cuando una ecuación con subíndice l tenga más monomios que un número fijado, se elimina l de los posibles subíndices. Es decir, a partir de ese momento dicho subíndice l será distinto de i, j_1 y j_2 en las sucesivas $I_i(j_1, j_2)$. En otras palabras, no se podrá añadir monomios nuevos a la ecuación l ni tampoco utilizarla para añadir monomios nuevos a otras ecuaciones. Esto asegura que el número de monomios en una ecuación no se dispare.

2. Los subíndices i_j de las aplicaciones I_{i_j}, se obtienen de manera aleatoria pero sin repetición del conjunto $\{1, \ldots, n\}$. Una vez se haya obtenido n subíndices repetimos el proceso y así sucesivamente. Esto es una medida para que el número de monomios en cada ecuación sea lo más parecido posible en cada iteración y manteniendo un procedimiento aleatorio.

3. Si solo queda un subíndice admisible, ya que las $n - 1$ ecuaciones restantes tienen más monomios de los deseados, el procedimiento termina.

Los experimentos muestran que no hay una dependencia clara entre el número i_k y la complejidad del F4, pero sí entre la cota máxima del número de monomios y la complejidad del F4. Por lo que, la elección de los parámetros es crucial para que el número de monomios no se dispare pero el sistema sea resistente al F4. Otro dato que hay que tener en cuenta es que la última triangular tiene la forma $I_i(j, j)$ lo que implica que haya una ecuación de la forma

$$y_i = x_i + x_j^2$$

Pero que el exponente sea par no supone un problema ya que, como ya se ha comentado antes, reiteradamente se hacen reducciones del grado ya que $x^q - x = 0$. Claramente el número de posibles sistemas que se puede generar de esta manera es muy elevado, por lo que no es viable un ataque de fuerza bruta que busque los parámetros iniciales con los que se ha generado el sistema. Como se muestra en la figura 6.2.1 con estas consideraciones podemos hacer que el número de monomios por ecuación no se dispare.

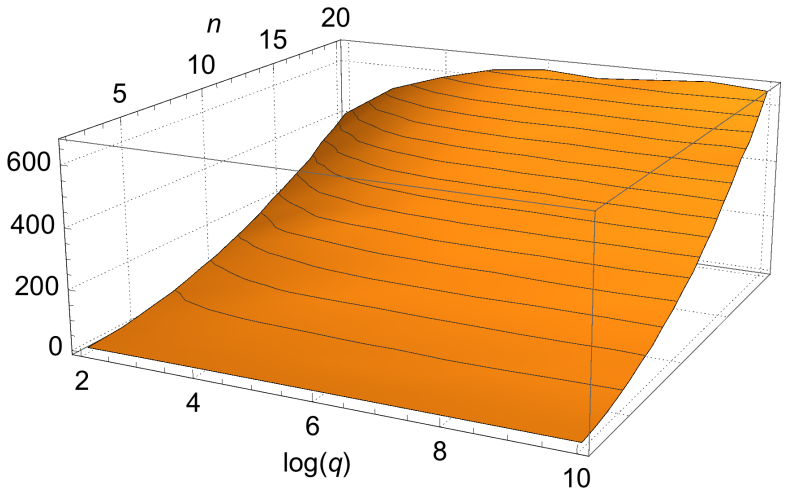

Figura 6.2.1: Número de monomios en función del cuerpo y del número de variables

Además, como ya se ha comentado antes, el tiempo de ejecución del F4 sobre este tipo de sistemas y sobre sistemas aleatorios es muy similar, por lo que el F4 no supone riesgo para este sistema. En los programas desarrollados he implementado el sistema en criptosistemaV3.1.sage para que se pueda probar.

Capítulo 7

Programas

1. Módulos

```
# -*- coding: utf-8 -*-
from sage.rings.all import ideal
from time import time
import numpy
#from polybori import *
from sage.rings.polynomial.toy_buchberger import *


def generar_anillo_polinomios(q,n,orden = 'degrevlex'):
    K.<a> = GF(q)
    P = PolynomialRing(K,'x',n, order=orden)     #Define el anillo de
        polinomios

    P.inject_variables(verbose=False)            #Define las variables del
        anillo para poder trabajar con ellas
    # X = P.gens_dict()                            #Crea un diccionario con
        las variables para poder trabajar con ellas
    # x = [0]*n
    # for i in range(0,n):
    #     xi = 'x%d' %i
    #     x[i] = X[xi]
    x = P.gens()                            #Definimos el vector x donde x[i] =
        xi para poder trabajar con las variables

    return K,P,x
```

```
def anadir_ecuaciones_cuerpo(I):
    x = I.ring().gens()                    #Definimos x,q,n a partir de I
    q = len(I.ring().base())
    n = len(x)

    J = []
    for j in range(0,n):                   #Genera las ecuaciones del
        cuerpo
        J.append(x[j]^q-x[j])              #Añadimos las euaciones
    I = I + ideal(J)

    return I


def groebner(I,informacion = 'no',algoritmo = 'singular:slimgb'):
    P = I.ring()

    t1 = time()
    B = I.groebner_basis(algoritmo)        #Calculamos la GB de nuestro ideal
    t2 = time()
    gradomax = max(f.degree() for f in B)      #se ve cual es el grado
        máximo en la base de Groebner calculada
    if informacion == 'si':
        print 'El tiempo ha sido de:',t2-t1, 'segundos.'
        print 'El grado máximo en la GB es:',gradomax

    return t2-t1,gradomax,B


def sistema_aleatorio(q,n,m,grado,solucion = 'si',homogeneo = 'no',monomios
    = 0,GB = 'si',informacion = 'si',ecuacionescuerpo = 'no'):
    if monomios == 0:                          #Si no se pone el número
        de monomios se considera como un sistema aleatorio
        monomios = binomial(n+grado,grado)

    K,P,x = generar_anillo_polinomios(q,n)     #Genera el anillo de
        polinomios

    if solucion == 'si':
        s = [K.random_element() for _ in range(n)]  #Genera un punto
            aleatorio del espacio afín para que sea la solucion del sistema

    if homogeneo == 'si':
        h = 'x%d' %n                           #Generamos variable para
```

```
            homogenizar

    L = []                                          #Lista de los polinomios
    for _ in range(m):                              #Genera un sistema aleatorio
         de polinomios homogéneos de un cierto grado
        f = P.random_element(degree=grado,terms=monomios)     #Genera un
            polinomio de grado 2 con un número determinado de términos
        if solucion == 'si':                            #Si se quiere que tenga
            solucion lo forzamos
            f -= f(*s)                                  #Modifica el polinomio
                generado restándole el valor de evaluar dicho polinomio en el
                 punto generado previamente
        if homogeneo == 'si':
            f = f.homogenize(h)                         #Homogeneizamos el
                sistema
        L.append(f)
    I = Ideal(P,L)


    if  ecuacionescuerpo == 'si':                   #Añadimos las ecuaciones del
         cuerpo
        I = anadir_ecuaciones_cuerpo(I)


    if GB == 'si':                                  #Hallamos una base de
        groebner en Función de los parámetros
        if informacion == 'si':
            print 'Sistema aleatorio:'
        tiempo,gradomax,B = groebner(I,informacion)
    else:
        tiempo,gradomax,B = 0,0,0

    return I,P,K,tiempo,gradomax,B


def serie_Hilbert(I):
    P = I.ring()                                                    #Generamos P
         y x a patir de I
    x = P.gens()

    if not I.is_homogeneous():                                      #Comprueba
        si es homogeneo
        L = [f.homogenize('x%d' %len(x)) for f in I.gens()]         #
            Homogeneizamos los polinomios
```

```
    I = Ideal(L)

TT.<t> = LaurentSeriesRing(ZZ)                    #Se define el anillo de
    las series de Laurenzt.
HS = I.Hilbert_series()(t)                        #Calcula la serie de
    Hilbert del ideal I, (si se ha homogeneizado podría salir que no es
    regular, o si tiene sulucion)
HS1 = I.Hilbert_series()                          #Calcula la serie de
    Hilbert

if HS.laurent_polynomial() == HS1:                #Comprueba si realemnte
    la serie de Hilbert es un polinmio
    listcoefi = HS.coefficients()                 #Generamos los
        coeficientes de la serie.
    cont = 0
    while ind == 0:                               #Buscamos el índice
        y salimos del bucle una vez encontrado.
        if listcoefi[cont] <= 0:
            ind = cont
            break
        if cont == len(listcoefi)-1:              #Por si la serie
            resulta ser un polinomio con todos los coeficientes positivos
            .
            ind = cont + 1
            break

        cont += 1
```

2. Programa dreghfe.sage

```
##-------------------------------------------------------------------------
##  -*- coding: utf-8 -*-
##  dreghfe.sage
##
##
##  Created by Jorge Linde on 2017.
##  Copyright (c) 2018 Jorge Linde. All rights reserved.
##-------------------------------------------------------------------------
#Programa con el cual se puede obtener los tiempos de ejecución necesarios
    para resolver el HFE en función de ciertos parámetros


# Módulos
# -------------------------------------------------------------------------
from sage.rings.all import ideal
from time import time
import numpy


# Funciones
# -------------------------------------------------------------------------
def dreghfe(D,q,n):                                 #Función principal
    #n es el número de variables, tamaño de la extensión del cuerpo grande
    #D máximo de la ecuación univariada
    #q tamaño del cuerpo

    mexp = D+1                                      #Acotamos el exponente
        máximo en los monomios cuadráticos del polinomio en una variable
    m1exp = D+1                                     #Acotamos el exponente
        máximo en los monomios lineales del polinomio en una variable
    K.<a> = GF(q)                                   #F_q
    R.<b> = PolynomialRing(K)                       #Generamos el anillo de
        polinomios para luego hacer el cociente, b será la base
    f = R.irreducible_element(n,algorithm='primitive')  #Generamos el
        polinomio característico de nuestra extensión


    def iso(v,n):                                   #Isomorfismo de (F_q)^n a
        F_q^n
        V = sum(v[i]*b^i for i in range(0,n))
        return V
```

```
    def isoinverse(V,n):                          #Isomorfismo de F_q^n a (F_q
        )^n
        V = V.quo_rem(f)[1]                       #Se reduce por si acaso no
            estuviese reducido
        v = [0]*n
        for i in range(0,n):
            v[i] = (V.quo_rem(b^(i+1))[1]).quo_rem(b^i)[0]
            V = V - v[i]*b^i
        return v


# Generación del entorno de trabajo
# ------------------------------------------------------------------------
    P = PolynomialRing(K,'x',n, orden='degrevlex')    #Define el anillo de
        polinomios
    PP.<b> = PolynomialRing(P)                    #Anade la variable b al
        anillo de polinomios P
    P.inject_variables(verbose=False)             #Define las variables del
        anillo para poder trabajar con ellas

    X = P.gens_dict()                             #Crea un diccionario con las
         variables para poder trabajar con ellas
    x = [0]*n
    for i in range(0,n):
        xi = 'x%d' %i
        x[i] = X[xi]                              #Definimos el vector x donde
             x[i] = xi para poder trabajar con las variables
    vx = matrix(P,x).transpose()                  #vx es el vector x
        transpuesto

    V = VectorSpace(K,n)                          #Generamos el espacio
        vectorial (F_q)^n
    A = [[] for _ in range(n)] for _ in range(n)]  #Generamos una matriz
        triangular superior donde almacenamos los elementos Aij
    for i in range(0,min(mexp,n)):
        for j in range(i,min(mexp,n)):
            aij = V.random_element()              #Generamos un elemento
                aleatorio de V
            A[i][j] = aij


# Generación del sistema HFE
# ------------------------------------------------------------------------
    sistema = [0]*n                               #Generamos el vector donde
```

```
    vamos a guardar las ecuaciones del sistema
Q = [[] for _ in range(n)] for _ in range(n)]  #Primero se calcula la
    matriz asociada a la aplicación lineal x^q. Calculamos sus columnas
for k in range(n):
    columna = isoinverse((b^k)^q,n)
    for z in range(n):
        Q[z][k] = columna[z]
Q = matrix(P,Q)                                #Generamos la matriz con
    coeficientes en P

for i in range(n):
    for j in range(n):
        if A[i][j]:                            #se hará operaciones si el
            elemento es no nulo
            product = isoinverse(iso(A[i][j],n)*iso(Q^i*vx,n)*iso(Q^j*vx
                ,n),n)  #Generamos los monomios a_ij*x^q^i*x^q^j
            for s in range(n):
                sistema[s] += product[s]   #Añadimos las ecuaciones a
                    nuestro sistema


B = [[] for _ in range(n)]                     #Generamos una matriz
    triangular superior donde almacenamos los elementos Bi,
for i in range(0,min(m1exp,n)):
    bi = V.random_element()
    B[i] = bi
for i in range(n):
    if B[i]:                                   #se hará operaciones si el
        elemento es no nulo
        product2 = isoinverse(iso(B[i],n)*iso(Q^i*vx,n)[0],n)        #
            Generamos los monomios b_i*x^q^i
        for s in range(n):
            sistema[s] += product2[s]      #Añadimos las ecuaciones a
                nuestro sistema



p = [K.random_element() for _ in range(n)]  #Generamos un punto
    aleatorio para que sea solución de nuestro sistema
SISTEMA = [P(sistema[i])-P(sistema[i])(p) for i in range(n)]
I = ideal(SISTEMA)                             #Generamos el ideal asociado
     a nuestro sistema

J = []
```

```
    if 0 == 0:                                  #Condicion para añadir las
        ecuaciones del cuerpo al ideal, si el q es pequeño la complejidad es
        menor
        for j in range(0,n):                    #Genera las ecuaciones del
            cuerpo
            J.append(x[j]^q-x[j])
        I = I + ideal(J)

    t11 = time()
    B = I.Groebner_basis('singular:slimgb')     #Calculamos la GB de nuestro
         ideal
    t21 = time()
    print 'El tiempo ha sido de:',t21-t11, 'segundos.'
    gradomax = max(f.degree() for f in B)       #se obtiene el grado máximo
        en la base de Groebner calculada
    print 'El grado máximo es:',gradomax
    return t21-t11                              #Tiempo necesario para
        calcular la base de Groebner


def random(q,n):                                #Programa que genera un
    sistema aleatorio y calcula su GB
    k.<a> = GF(q)                               #Define el cuerpo base sobre
         el que trabajar
    Q = PolynomialRing(k,'y',n, orden = 'degrevlex')    #Define el anillo de
         polinomios
    Q.inject_variables(verbose = False)         #Define las variables.
        verbose = False para que no lo muestre por pantalla
    Y = Q.gens_dict()                           #Crea un diccionario con las
         variables para poder trabajar con ellas
    y = [0]*n
    for i in range(0,n):
        yi = 'y%d' %i
        y[i] = Y[yi]
    s = [k.random_element() for _ in range(n)]  #Genera un punto aleatorio
        del espacio afín
    L = []                                      #Lista de los polinomios
    for w in range(n):                          #Genera un sistema aleatorio
         de polinomios homogéneos de un cierto grado
        f = Q.random_element(degree=2,terms=binomial(n+2,2))    #Genera un
            polinomio de grado 2 con nu número determinado de términos
        f -= f(*s)                              #Modifica el polinomio
            generado restándole el valor de evaluar dicho polinomio en el
            punto generado previamente
```

```
        L.append(f)
    if 0 == 0:                                  #Condicion para añadir las
        ecuaciones del cuerpo
        for j in range(0,n):                    #Genera las ecuaciones del
            cuerpo
            L.append(y[j]^q-y[j])

    I = Ideal(L)                                #Genera el ideal asociado a
        la lista de polinomios
    t1 = time()
    g = I.Groebner_basis('singular:slimgb')
    t2 = time()
    print 'El tiempo de ejecución en el sistema aleatorio ha sido:',t2-t1
    gradomax = max(f.degree() for f in g)
    print 'El grado máximo es para el sistema aleatorio es:',gradomax
    print ' '
    return t2-t1


# Parámetros
# ------------------------------------------------------------------------
q = 2                                           #Tamaño del cuerpo
n = 5                                           #número de variables inicial
vn = 18                                         #Rango en el que varía el
    número de incógnitas, de n a n+vn
media = 5                                       #número de veces que se hará
     el calculo para obtener la media
máximo = 9                                      #Grado máximo de los
    polinomios (q^máximo)
set_verbose(0)                                  #Cantidad de información que
     nos genera Sage
DS = range(1,máximo+1)
T = matrix(RR,vn,máximo+1)                      #Matriz donde se guardara la
     información


# Programa
# ------------------------------------------------------------------------
for j in range(vn):
    for D in DS:
        l = 0                                   #En esta variable se
            guardara el tiempo de ejecución medio
        for _ in range(media):
            ls = dreghfe(D,q,n+j)
```

```
            l += float(ls)/float(media)
        T[j,D-1] = l

    l = 0
    for _ in range(media):
        ls = random(2,n+j)
        l += float(ls)/float(media)
    T[j,máximo] = l

nombre = 'Datos/dreghfe%d%d.csv' %(vn,máximo)  #Genera los nombre de los
    archivos donde se van a guardar los datos
numpy.savetxt(nombre, T, delimiter=',',fmt = '%f')  #Guardamos los datos
    para luego poder visualizarlos con el mathematica
print T
```

3. Programa isregular.sage

```
##-------------------------------------------------------------------------
##  -*- coding: utf-8 -*-
##  isregular.sage
##
##
##  Created by Jorge Linde on 2018
##  Copyright (c) 2018 Jorge Linde. All rights reserved.
##-------------------------------------------------------------------------
#Programa con el cual se puede obtener el porcentaje de sistemas que son
    regulares en Función de ciertos parámetros


# Módulos
# -------------------------------------------------------------------------
from time import time
import numpy
load("modulos.sage")
set_verbose(0)                                  #Cantidad de informacion
     que se muestra por pantalla


# Funciones
# -------------------------------------------------------------------------
def isregular(n, maxdeg, q):
    salida = 0
    nvar = n-1                                  #número de variables
        menos la variable homogenea
    i = ZZ.random_element(n,2*n)                #número de ecuaciones
        que va a tener nuestro sistema aleatorio
    j = ZZ.random_element(2,maxdeg+1)           #Grado aleatorio en un
        rango que va a tener nuestro sistema aletaorio
    k,P,x = generar_anillo_polinomios(q,nvar,orden = 'degrevlex')
    h = 'x%d' %nvar                             #Variable con la cual se
         va a homogenizar.
    R.<t> = LaurentSeriesRing(ZZ)               #Se define el anillo de
        las series de Laurenzt.

    hs = (((1-t^j)/(1-t))^n*(1-t^j)^i)          #Serie de Hilbert para
        una sucesion semi-regular.
    ind = 0                                     #Esta variable guardara
        el índice del primer coeficiente no negativo (el índice de
        regularidad)
    listcoefi = hs.coefficients()               #Generamos los
```

```
        coeficientes de la serie.
    cont = 0

    while ind == 0:                                  #Buscamos el índice y
        salimos del bucle una vez encontrado
        if listcoefi[cont] <= 0:
            ind = cont
            break

        if cont == len(listcoefi)-1:                 #Por si la serie resulta
             ser un polinomio con todos los coeficientes positivos.
            ind = cont + 1
            break
        cont += 1
    hs = hs.truncate(ind)                            #Trunca la serie para
        que sea la serie de una sucesion semi-regular.

    L = []                                           #Lista de los polinomios
        .
    for w in range(n+i):                             #Genera un sistema
        aleatorio de polinomios homogeneos de un cierto grado fijo como
        máximo.
        f = P.random_element(degree = j,terms = binomial(nvar+j,j))
        L.append(f.homogenize(h))                    #Anade el polinomio
            homogeneizado a la lista.
    I = Ideal(L)
    HS = I.Hilbert_series()                          #Calculamos la HS del
        ideal aleatorio generado.

    if HS == hs:                                     #Comprobamos si la serie
         es semi-regular.
        salida = 1
    return salida


# Parámetros
# ------------------------------------------------------------------------
N = []                                               #Genera la matriz donde
    se va a guardar los poercentajes.
minn = 3
maxn = 10
maxdeg = 2
tamanos = [2,3,4,5,7]                                #Tamano de los cuerpos.
char = 2
```

```
itera = 30.0
N = matrix(QQ,maxn-minn,len(tamanos))
coni = 0


# Programa
#-----------------------------------------------------------------------
for n in range(minn,maxn):                          #i es el número de
    ecuaciones extras.
    conj = 0
    for char in tamanos:
        porcentaje = 0                              #Variable que nos guarda
             el porcentaje de sucesiones semi-regulares.
        for K in range(itera):                      #Genera itera sistemas
            para hacer la media.
            V = isregular(n,maxdeg,char)
            porcentaje += V

        N[coni, conj] = porcentaje*100.0/itera      #Guardamos el porcentaje
            .
        print n,char
        conj += 1
    coni +=  1

nombre = 'isregular%d%d%d.csv' %(maxn,maxdeg,len(tamanos)) #Crea el nombre
    con el que se va a guardar los datos.
print N
numpy.savetxt(nombre, N, delimiter=',',fmt = '%f')
show(list_plot3d(N))
```

4. Programa carldreg.sage

```
##-------------------------------------------------------------------------
##  -*- coding: utf-8 -*-
##  caldreg.sage
##
##
##  Created by Jorge Linde on 2018
##  Copyright (c) 2018 Jorge Linde. All rights reserved.
##-------------------------------------------------------------------------
#Programa que calcula el índice de regularidad para sistemas cuadraticos de
    n variables y 2n ecuaciones.Donde la n varía entre 2 y max+2


# Módulos
# -------------------------------------------------------------------------
import numpy                                    #Para utilizar las funciones
    de guardado.
from time import time
import numpy as np


# Parámetros
# -------------------------------------------------------------------------
t1 = time()                                     #Función para medir el
    tiempo.
opcion = 1                                      #Valor que determina de que
    manera se va a calcular los coeficientes de la serie, la más optima es la
     opcion 1.
max = 5000                                      #Valor máximo para el número
    de iteraciones, valor máximo para el número de variables.
N = [[1,2]]                                     #Variable donde se guardaran
    los datos.
cocientes = [0]


# Programa
# -------------------------------------------------------------------------
i = var('i')                                    #Hay que definir la variable
    en la que se suma.
opt = 1                                         #El dreg va aumnetando por
    lo que los primeros no hace falta calcularlos.
for n in range(2,max+3,100):
    t = 0                                       #Se utiliza para salir del
        bucle y encontar el primer coeficiente no positivo.
```

```
    for k in range(opt-1,n+1):                  #Emepezamos en grado 1 0=opt
        . Se podría empezar directamente desde opt ya que la Función es
        monotona creciente
        if opcion == 0:
            a = sum((-1)**i*binomial(n, i)*binomial(2*n, k-i), i, 0, k)
        else:
            parcial = binomial(2*n,k)           #Otra manera de calcular lo
                mismo que la opcion 0, pero esta manera es mucho más
                eficiente ya que evita hacer calculos que ya se ha hecho.
            sumatorio = parcial
            for i in range(0,k):                #La fórmula de cada
                coeficiente de la serie viene determianda por la anterior
                expresion de a. Mirar la teoria en cualquier caso para n var
                y 2n ecu.
                parcial = parcial*(n-i)*(k-i)/(i+1)/(2*n-k+i+1)*(-1)
                sumatorio = sumatorio + parcial
        a = sumatorio                           #a es el coeficiente c_k.


        if (t == 0) and (a <= 0):               #Si t == 0 (no se ha
            encontrado antes otro coeficiente no positivo) y el coeficiente
            es no positivo entonces guardalo.

            if N[len(N)-1][1] > k:              #Comprueba si en estas
                condiciones la Función dreg(n) no es monotona creciente.
                print 'error'
            N.append([n,k])                     #Anade los parámetros
            if n == 1:
                cocientes.append(0)             #Se guardan los cocientes
                    entre el índice de regularidad y el número de variables.
            else:
                cocientes.append(float(k-2)/float(n-1))
            t = 1                               #Ya se ha encontrado el
                primero coeficiente no positivo.
            opt = k                             #A partir de ahora
                epezaremos por este coeficiente, ya que a medida que
                aumentamos la n el índice de regularidad aumenta o se queda
                igual(esto no esta demostrado).
            print n
            break                               #Con esta opcion, break, no
                haria falta considerar la variable t.

t2 = time()
print t2-t1
```

```
print N
N2 = np.array(N)
print N2[:,1]
print cocientes
nombre = 'listadreg%d.csv' %max
numpy.savetxt(nombre, N, delimiter=',',fmt = '%f') #Mejor para el
    mathematica, guarda los datos obtenidos para representarlos en el
    mathematica.
```

5. Programa caldreggen.sage

```
##-------------------------------------------------------------------------
##  -*- coding: utf-8 -*-
##  caldreggen.sage
##
##
##  Created by Jorge Linde on 2018
##  Copyright (c) 2018 Jorge Linde. All rights reserved.
##-------------------------------------------------------------------------
#Programa que calcula el índice de regularidad en función del número de
    ecuac y variables para luego guardarlo en una matriz. Para polinomios
    cuadraticos


# Módulos
#--------------------------------------------------------------------------
import numpy
                                                                    #Esta
    función sirve para importar la matriz.
from time import time
load('modulos.sage')


# Parámetros
#--------------------------------------------------------------------------
t1 = time()
opcion = 1
                                                                        #
    Dependiendo del valor calcula el dreg de una forma u otra (La mejor la
    opción 1).
nmax = 150
                                                                        #
    Valor máximo para n.
mmax = 2*nmax
                                                                    #Valor
    máximo para m.
N = []
N = matrix(ZZ,nmax,mmax)
                                                                #Crea una matriz
    donde se guardaran los índices de regularidad en función del m y n.
C = matrix(QQ,nmax,mmax)
cocientes = [0]
i = var('i')
                                                                    #Hay
```

```
    que definir la variable en la que se suma en los binomios.


# Programa
# ------------------------------------------------------------------------
for n in range(1,nmax+1,1):
                                                        #En este rango la n
    esta en el intervalo [1,nmax].
    mmaxreal = min(mmax+1,binomial(n+1,2)+1)
                                            #Este es el máximo real de
        ecuacacuiones que puede tener un sistema regular de n variables y
        cuadratico.
    opt = 0
                                                                          #
        El dreg va aumnetando cuando la n disminuye, por lo que los primeros
        no hace falta calcularlos.
    for m in range(mmaxreal-1,n-1,-1):
                                                        #En este rango la m
        esta en el intervalo [n,2n] o [n,mmax].
        for k in range(opt,2*m-n+2):
                                                        #El índice de
            regularidad es monotono creciente, cuando la m disminuye. El
            valor máximo que puede tomar es 2m-n.
            #if opcion == 0:
                #a = sum((-1)^i*binomial(m-n,i)*binomial(m,k-i),i,max(0,k-m)
                    ,min(k,m-n))
            #else:
            imin = max(0,k-m)
                                                                #Intervalo
                en el que se va a sumar la i.
            imax = min(k,m-n)


            parcial = binomial(m,k-imin)
                                                        #De esta manera se
                calcula la suma de los binomios de una manera más eficiente.
            sumatorio = parcial*(-1)^(imin)
            for i in range(imin,imax):
                parcial = parcial*(m-n-i)*(k-i)/(i+1)/(m-k+i+1)*(-1)
                sumatorio = sumatorio + parcial
            a = sumatorio

            if a <= 0:
                                                                          #
```

```
                Comprueba si el coeficiente k-esimo es negativo, y si es así
                lo guarda en la matriz.
                N[n-1,m-1] = k
                C[n-1,m-1] = float(k)/float(n)
                                                    #Guarda una matriz el
                    cociente entre k y n.
                opt = k
                                                                        #
                    El índice de regularidad va aumentando a medida que el
                    número de ecauaciones disminuye.
                break

                    #Cuando se calcula el dreg no hace falta seguir
                    calculando monomios.
    print float(n)/float(nmax)*100,'%

t2 = time()
print N
print C

numpy.savetxt('listadreggen.csv', N, delimiter=',',fmt = '%f')
                    #Guarda los datos en unas matrices para luego leerlos
    con el matematica.
numpy.savetxt('listadreggenC.csv', C, delimiter=',',fmt = '%f')
tiempo = int(t2-t1)
horas = tiempo//3600
                                                                #Muestra el
    tiempo empleado para hacer los calculos.
minutos = (tiempo%3600)//60
segundos = tiempo%60
print 'Se ha tardado: ',horas,' horas ',minutos,' minutos ',segundos,'
    segundos.'
```

6. Programa criptosistemaV3.1.sage

```
##-------------------------------------------------------------------------
##  -*- coding: utf-8 -*-
##  criptosistemaV3.1.sage
##
##
##  Created by Jorge Linde on 2018
##  Copyright (c) 2018 Jorge Linde. All rights reserved.
##-------------------------------------------------------------------------
#Programa que nos genera sistemas MPKC basado en la composición de
    aplicaciones triangulares. Además, nos muestra cierta informacion extra
    de los sistemas


# Módulos
#-------------------------------------------------------------------------
from time import time
from sage.combinat.permutation import Permutations
from sage.matrix.matrix_space import MatrixSpace
from sage.rings.all import ideal
import numpy
load("modulos.sage")


# Funciones
#-------------------------------------------------------------------------
def criptosistema(l ,n,pasos,ind):
    q = 2**l
    k,P,x = generar_anillo_polinomios(q,n,orden = 'degrevlex')
    J = []
    for j in range(0,n):                                    #Genera las
        ecuaciones del cuerpo
        J.append(x[j]^q-x[j])
    I = ideal(J)


    def matrizaleatoria(k,n,m):                             #Esta Función genera
         una matriz aleatoria en el anillo k de dimensión n*m
        M = MatrixSpace(k,n,m)                              #Genera el espacio
            de matrices
        A = M.random_element()                              #Elige una al azar
        while A.rank() < min(n,m):                          #Comprueba que tiene
             orden máximo
            A = M.random_element()
```

```
    return A


def triangular_aleatoria(ind,posicion,lista):    #Función que nos
    devuelve los datos necesarios para generar un termino. No tiene
    termino independiente
    y = [0]*n
    listatemp = list(set(lista).difference({posicion})) #Generamos la
        lista con los posibles subíndices
    possub = Permutations(listatemp)             #Generamos el conjunto
        de permutaciones de los subíndices posibles
    for i in range(n):
        y[i] = x[i]
    jind = [0]*ind
    for i in range(ind):                         #Genera los subíndices
        de los monomios que van a tomar parte en el termino a anadir
        while True:
            jind[i] = (possub.random_element())[0]
            if jind[i] != posicion:              #Se comprueba que dichos
                subíndices sean distintos de la fila a la que le vamos a
                anadir el termino
                break
    while True:
        coef = k.random_element()                #Generamos el
            coeficiente distinto de 0
        if coef != 0:
            break
    return [jind,coef]                           #Devolvemos los datos
        necesarios para anadir el termino en cuestion


def expmatri(A, v):                              #Función exponencial
    return [prod(v[j]**A[i,j] for j in range(0,n)) for i in range(0,n)]


def promatri(A, v):                               #Función que nos
    devuelve el producto de una matriz por un vector
    return [sum(v[j]**A[i,j] for j in range(0,n)) for i in range(0,n)]


def producto_lineal(A, S):                       #Función que nos
    devuelve el producto de una matriz por un sistema de ecuaciones
    return [sum(S[j]*A[i,j] for j in range(0,n)) for i in range(0,A.
        nrows())]
```

```
def composición(S1, S2):                    #Función que nos
    devuelve la composición de dos sistemas de ecuaciones
    S3 = [0]*n
    for i in range(n):
        S3[i] = S1[i].substitute({x[j]:S2[j] for j in range(n)})
    return S3


def componer_triangular(T, posicion, sistema, ind):        #Función que
     genera la composicon de triangulares
    monomios_elevados = []
    for k0 in range(ind):
        f = sistema[T[0][k0]]
        C = f.coefficients()
        M = f.monomials()
        E = 2^(ZZ.random_element(0, 1, 'uniform'))
        T_E = sum((C[i0]*M[i0])^E for i0 in range(len(C)))
        monomios_elevados.append(T_E)
    sistema[posicion] += T[1]*prod(monomios_elevados[i0] for i0 in range
        (ind))
    return sistema


def construccion(pasos,ind):
    lista = range(0,n)
        #Genera la lista inicial de los subíndices admisibles
    listapos = Permutations(lista).random_element()
        #Genera una permutacion aleatoria de la lista
    sistema = x
        #Quitamos la exponencial, se empieza con x_i = x_i
    contador = 0
        #Utizaremos este contador para saber cual es la siguiente fila a
        la que se le anadiran los monomios
    contador_pasos = 0
        #Este contador lleva el número total de pasos ejecutados
    while True:
        T = triangular_aleatoria(ind,listapos[contador],lista)
        sistema = componer_triangular(T,listapos[contador],sistema,ind)
        sistema[listapos[contador]] = sistema[listapos[contador]].reduce
            (I.gens())  #Reduce las ecuaciones del sistema por las
            ecuaciones del cuerpo para bajar los grados
```

```
        if len(sistema[listapos[contador]].monomials()) > pasos:
                                #Comprueba si el número de monomios es
            mayor que el deseado
            lista = list(set(lista).difference(set([listapos[contador]])
                ))          #En dicho caso elimina el subíndice
                correspondiente de la lista de subíndices admisibles

        if 0 == 1:                                                  #Con
             este comando pintariamos el sistema en cada iteracion
            for i1 in range(n):
                print ''
                print 'y[%d]=' %i1 ,sistema[i1]
        contador += 1                                               #
            Avanzamos un paso
        contador_pasos += 1
        if contador >= len(listapos):                               #Si
            nos se pasa en pieza de cero con una nueva lista de
            posiciones
            contador = 0
            listapos = Permutations(lista).random_element()
        if len(lista) == 1:
            if 0 == 0:
                for i in range(n):                                  #
                    Para ver los polinomios del core antes de la
                    aplicación lineal
                    print ''
                    print 'y[%d]=' %i ,sistema[i]
            break
    número_monomios = [len(sistema[l1].exponents()) for l1 in range(n)]
    return sistema,contador_pasos,número_monomios


def randomice(sistema):                                             #Con
     esta Función modificamos el sistema añadiendo monomios aleatorios
    sistema_list = ideal(sistema).gens()
    A_1,monomios = sistema_list.coefficient_matrix()                #
        Otra manera de obtener los monomios

    for i in range(n):
        coeficientes = matrix([k.random_element() for _ in range(
            monomios.nrows())])
        fr = coeficientes*monomios
        sistema.append(fr)
    return sistema
```

```
def evaluacion_rapida(sistema,punto):
    sistema_list = ideal(sistema).gens()
    A,v = sistema_list.coefficient_matrix()
    for i in range(v.nrows()):
        v[i] = v[i](*punto)
    evaluacion1 = A*v
    evaluacion = [j[0] for j in evaluacion1]
    return evaluacion


sistema, contador_pasos, número_monomios = construccion(pasos,ind)  #
    Sistema sin las ecuaciones aleatorias
print 'Sistema generado.'
sistemaran = randomice(sistema)                                      #Se
    anaden las ecuaciones aleatorias al sistema
print 'Sistema aleatorio generado.'
sistema = producto_lineal(matrizaleatoria(k, n, n),sistema)
print 'Producto lineal sistema generado.'
sistemaran = producto_lineal(matrizaleatoria(k,2*n,2*n),sistemaran)
print 'Producto lineal sistema ran generado.'

s = [k.random_element() for _ in range(n)]

eval1 = time()
evaluacion = evaluacion_rapida(sistemaran,s)
for i in range(2*n):                                                 #se
    hará que el sistema tenga solucion, (el texto claro)
    sistemaran[i] -= evaluacion[i]
eval2 = time()                                                       #
    Tarda mucho en evaluar
print 'El tiempo de evaluacion ha sido optimizado es: ',eval2-eval1

if 0 == 1:                                                           #Con
     esta Función elegimos si quieremos que nos muestre el sistema por
    pantalla
    print ''
    print ''
    for i in range(n):
        print ''
        print 'y[%d]=' %i ,sistemaran[i]
return sistema,sistemaran,contador_pasos,número_monomios
```

```
# Parámetros
# ------------------------------------------------------------------------
nmax = 15                       #Valor máximo para el n
pasosmax = 30                   #número máximo de composiciones
ind = 2                         #número de terminos en la triangular
iteraciones = 1                 #número de sistemas que se van a generar
Nmono = matrix(QQ, iteraciones ,2)
Nvari = matrix(QQ, iteraciones ,2)
Npasos = matrix(QQ, iteraciones ,2)
pasosmedia = 1
lm = 4                          #Falta ver si hay diferencia en la complejidad
    por la diferencia de tamano
monmin = 20                     #número minimo de monomios
monmax = monmin + 1


# Programa
# ------------------------------------------------------------------------
for n in range(nmax,nmax+1):
    for i3 in range(iteraciones):
        t2 = time()
        t1 = time()
        #l = ZZ.random_element(x = 2, y = lm+1, distribution = 'uniform')
        l = lm
        pasos = ZZ.random_element(x = monmin, y = monmax, distribution = '
            uniform')                   #número de monomios
        media_monomios = 0
        tpeval = 0
        dif1 = 0
        for _ in range(pasosmedia):
            sistema,sistemaran,contador_pasos,número_monomios =
                criptosistema(l,n,pasos,ind)
            media_monomios += sum(len(sistema[l1].monomials()) for l1 in
                range(n))               #Ponemos 2 para que nos coja solo
                las dos primeras filas
            tpeval += dif1
        tpeval = float(tpeval)/float(pasosmedia)
        media_monomios = float(media_monomios)/float(pasosmedia*n)
        media = float(sum(número_monomios[i0] for i0 in range(n)))/n
        varianza = (float(sum((número_monomios[i1]-media)^2 for i1 in range(
            n)))/n)^(1/2)
        print i3,': Para n = ',n,', q = ',2^l,'e pasos = ',pasos,' el número
            de monomios medio es: ',media_monomios,' y tamano: ',float(n*
```

```
            media*l*2)/float(8000),'KBytes'
        print 'El número de iteraciones es: ',contador_pasos,', los tamanos
            iniciales son: ',número_monomios,' y varianza: ',varianza

        if 0 == 1:                          #Por si querermos calcular la base
            de Groebner
            I = Ideal(sistema)                                          #Genera el
                 ideal asociado a la lista de polinomios
            t1 = time()
            g = I.groebner_basis('singular:slimgb')
            t2 = time()
            gradomax = max(f.degree() for f in g)
            print 'El tiempo del F4 para el sistema ha sido: ',t2-t1,' con un
                grado máximo: ',gradomax
        if 0 == 1:                          #Por si querermos calcular la base
            de Groebner
            I = Ideal(sistemaran)                                       #Genera
                 el ideal asociado a la lista de polinomios
            t1 = time()
            g = I.groebner_basis('singular:slimgb')
            t2 = time()
            gradomax = max(f.degree() for f in g)
            print 'El tiempo del F4 para el sistemaran ha sido: ',t2-t1,' con
                un grado máximo: ',gradomax
            print' '
            Nmono[i3] = [media_monomios, t2-t1]
            Nvari[i3] = [varianza, t2-t1]
            Npasos[i3] = [l, t2-t1]

nombre1 = 'monomiossistemaV3%d%d%d%d.csv' %(nmax,iteraciones,lm,monmax)
    #Crea el nombre con el que se va a guardar los datos.
nombre2 = 'varianzasistemaV3%d%d%d%d.csv' %(nmax,iteraciones,lm,monmax)
nombre3 = 'pasossistemaV3%d%d%d%d.csv' %(nmax,iteraciones,lm,monmax)
numpy.savetxt(nombre1, Nmono, delimiter=',',fmt = '%f')
numpy.savetxt(nombre2, Nvari, delimiter=',',fmt = '%f')
numpy.savetxt(nombre3, Npasos, delimiter=',',fmt = '%f')
```

Bibliografía

[AA10] Martin Albrecht and Martin Albrecht. *Algorithmic Algebraic Techniques and their Application to Block Cipher Cryptanalysis.* PhD thesis, 2010.

[AFI+04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner basis algorithms. In *Advances in cryptology—ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Comput. Sci.*, pages 338–353. Springer, Berlin, 2004.

[AM69] Michael Francis Atiyah and I. G. MacDonald. *Introduction to commutative algebra.* Addison-Wesley-Longman, 1969.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Pierre et Marie Curie-Paris VI, 2004.

[Bar07] Gregory V. Bard. *Algorithms for solving linear and polynomial systems of equations over finite fields, with applications to cryptanalysis.* ProQuest LLC, Ann Arbor, MI, 2007. Thesis (Ph.D.)–University of Maryland, College Park.

[Bar09] Gregory V. Bard. *Algebraic Cryptanalysis.* Springer Publishing Company, Incorporated, 1st edition, 2009.

[BBBV96] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *Submitted to: SIAM J. Sci. Statist. Comput.*, 1996.

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography.* Springer-Verlag, Berlin, 2009.

[BCC+13] Charles Bouillaguet, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, and Bo-Yin Yang. Fast exhaustive search for quadratic systems in $\mathbb{F}_2$ on fpgas — extended version. 2013.

[BcFSyY] M. Bardet, J. c. Faugère, B. Salvy, and B y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *IN MEGA'05, 2005. EIGHTH INTERNATIONAL SYMPOSIUM ON EFFECTIVE METHODS IN ALGEBRAIC GEOMETRY*.

[BDMM09] Johannes A. Buchmann, Jintai Ding, Mohamed Saied Emam Mohamed, and Wael Said Abd Elmageed Mohamed. Mutantxl: Solving multivariate polynomial equations for cryptanalysis. In *Symmetric Cryptography, 11.01. - 16.01.2009*, 2009.

[Ben80] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980.

[BFS03] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity of Gröbner basis computation for Semi-regular Overdetermined sequences over F_2 with solutions in F_2. Research Report RR-5049, INRIA, 2003.

[BFS04] Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *Journal of Symbolic Computation*, 70:49–70, September 2015.

[BFSY05] Magali Bardet, Jean-Charles Faugere, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems. In *The Effective Methods in Algebraic Geometry Conference (MEGA'05)(P. Gianni, ed.)*, pages 1–14, 2005.

[BKW12] T. Becker, H. Kredel, and V. Weispfenning. *Gröbner Bases: A Computational Approach to Commutative Algebra*. Springer New York, 2012.

[buc98] *Gröbner Bases and Applications*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1998.

[Cab10] Daniel Cabarcas. An implementation of faugère's f4 algorithm for computing gröbner bases. 2010.

[CG17] Alessio Caminata and Elisa Gorla. Solving multivariate polynomial systems and an invariant from commutative algebra. *CoRR*, abs/1706.06319, 2017.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in cryptology—EUROCRYPT 2000 (Bruges)*, volume 1807 of *Lecture Notes in Comput. Sci.*, pages 392–407. Springer, Berlin, 2000.

[CKR04] M. Caboara, M. Kreuzer, and L. Robbiano. Efficiently computing minimal sets of critical pairs. *Journal of Symbolic Computation*, 38(4):1169 – 1190, 2004. Symbolic Computation in Algebra and Geometry.

[CLO07] David Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms.* Undergraduate Texts in Mathematics. Springer, New York, third edition, 2007. An introduction to computational algebraic geometry and commutative algebra.

[CP03] Nicolas T. Courtois and Jacques Patarin. About the XL algorithm over GF(2). In *Topics in cryptology—CT-RSA 2003*, volume 2612 of *Lecture Notes in Comput. Sci.*, pages 141–157. Springer, Berlin, 2003.

[DG10] Vivien Dubois and Nicolas Gama. The degree of regularity of hfe systems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 557–576. Springer, 2010.

[DGS06] Jintai Ding, Jason E. Gower, and Dieter S. Schmidt. *Multivariate public key cryptosystems*, volume 25 of *Advances in Information Security.* Springer, New York, 2006.

[DH06] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[DHK+13] Jintai Ding, Timothy Hodges, Victoria Kruglov, Dieter Schmidt, and Stefan Tohaneanu. Growth of the ideal generated by a quadratic multivariate function over gf(3). 12, 05 2013.

[Dic92] Matthew Dickerson. The inverse of an automorphism in polynomial time. *J. Symbolic Comput.*, 13(2):209–220, 1992.

[Die04] Claus Diem. The xl-algorithm and a conjecture from commutative algebra. In *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2004.

[Die15] Claus Diem. Bounded regularity. *Journal of Algebra*, 423:1143 – 1160, 2015.

[DK] Jintai Ding and Thorsten Kleinjung. Degree of regularity for hfe-.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[Ede08a] C. Eder. On The Criteria Of The F5 Algorithm. *ArXiv e-prints*, April 2008.

[Ede08b] C. Eder. The Algorithmic Behaviour of the F5 Algorithm. *ArXiv e-prints*, October 2008.

[Ede13] Christian Eder. An analysis of inhomogeneous signature-based gröbner basis computations. *Journal of Symbolic Computation*, 59:21 – 35, 2013.

[EF14] C. Eder and J.-C. Faugère. A survey on signature-based Gr\"obner basis computations. *ArXiv e-prints*, April 2014.

[Eis95] D. Eisenbud. *Commutative Algebra with a view toward Algebraic Geometry*. Springer, Berlin, 1995.

[FA04] Jean-Charles Faugère and Gwénolé Ars. Comparison of XL and Gröbner basis algorithms over Finite Fields. Research Report RR-5251, INRIA, 2004.

[Fau99] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra*, 139(1-3):61–88, 1999. Effective methods in algebraic geometry (Saint-Malo, 1998).

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 75–83 (electronic). ACM, New York, 2002.

[Fel05] Adam Feldmann. A survey of attacks on multivariate cryptosystems. 2005.

[FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In *Advances in cryptology – CRYPTO 2003. 23rd annual international cryptology conference, Santa Barbara, California, USA, August 17–21, 2003. Proceedings*, pages 44–60. Berlin: Springer, 2003.

[GC00] Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances in cryptology—ASIACRYPT 2000 (Kyoto)*, volume 1976 of *Lecture Notes in Comput. Sci.*, pages 44–57. Springer, Berlin, 2000.

[GM88] Rüdiger Gebauer and H. Michael Möller. On an installation of buchberger's algorithm. *Journal of Symbolic Computation*, 6(2):275 – 286, 1988.

[Gro96] Lov K. Grover. A Fast quantum mechanical algorithm for database search. 1996.

[Hey13] Stefan Heyse. *Post quantum cryptography: implementing alternative public key schemes on embedded devices.* PhD thesis, PhD thesis, dissertation for the degree of doktor-ingenieur: 10.2013/Stefan Heyse.–Bochum, 2013.–235 p.–Bibliogr, 2013.

[HMS14] T. J. Hodges, S. D. Molina, and J. Schlather. On the Existence of Semi-Regular Sequences. *ArXiv e-prints*, December 2014.

[HPS14] Timothy J. Hodges, Christophe Petit, and Jacob Schlather. First fall degree and weil descent. *Finite Fields and Their Applications*, 30:155 – 177, 2014.

[HS13] Timothy J. Hodges and Jacob Schlather. The degree of regularity of a quadratic polynomial. *Journal of Pure and Applied Algebra*, 217(2):207 – 217, 2013.

[Kob98] Neal Koblitz. *Algebraic aspects of cryptography*, volume 3 of *Algorithms and Computation in Mathematics.* Springer-Verlag, Berlin, 1998.

[KR05] Martin Kreuzer and Lorenzo Robbiano. *Computational commutative algebra. 2.* Springer-Verlag, Berlin, 2005.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem. Springer, 1999.

[Kun12] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry.* Modern Birkhäuser Classics. Springer New York, 2012.

[Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer algebra (London, 1983)*, volume 162 of *Lecture Notes in Comput. Sci.*, pages 146–156. Springer, Berlin, 1983.

[Laz92] D. Lazard. Solving zero-dimensional algebraic systems. *Journal of Symbolic Computation*, 13(2):117 – 131, 1992.

[LN94] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, 2 edition, 1994.

[Lue18] Ignacio Luengo. https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions, 2018.

[Mat87] H. Matsumura. *Commutative Ring Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1987.

[MCY04] T. Moh, Jiun-Ming Chen, and Bo-Yin Yang. Building Instances of TTM Immune to the Goubin-Courtois Attack and the Ding-Schmidt Attack. *IACR Cryptology ePrint Archive*, (168), 2004.

[MH78] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in cryptology—EUROCRYPT '88 (Davos, 1988)*, volume 330 of *Lecture Notes in Comput. Sci.*, pages 419–453. Springer, Berlin, 1988.

[MMT92] H. Michael Möller, Teo Mora, and Carlo Traverso. GrÖbner bases computation using syzygies. In *Papers from the International Symposium on Symbolic and Algebraic Computation*, ISSAC '92, pages 320–328, New York, NY, USA, 1992. ACM.

[Moh99] T. Moh. A public key system with signature and master key functions. *Comm. Algebra*, 27(5):2207–2222, 1999.

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[NIS] NIST. https://csrc.nist.gov/publications/detail/nistir/8105/final.

[NMM+14] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt. Quantum computations on a topologically encoded qubit. *Science*, 2014.

[Pap] Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK.

[Par10] Keith Pardue. Generic sequences of polynomials. *Journal of Algebra*, 324(4):579 – 590, 2010.

[Pat96a] Jacques Patarin. Asymmetric cryptography with a hidden monomial. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 45–60, London, UK, UK, 1996. Springer-Verlag.

[Pat96b] Jacques Patarin. Asymmetric cryptography with a hidden monomial and a candidate algorithm for $\simeq$ 64 bits asymmetric signatures. In *Advances in cryptology—CRYPTO '96 (Santa Barbara, CA)*, volume 1109 of *Lecture Notes in Comput. Sci.*, pages 45–60. Springer, Berlin, 1996.

[Pat96c] Jacques Patarin. Hfe first challenge. 1996.

[Pat96d] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Advances in Cryptology—Eurocrypt'96*, pages 33–48. Springer, 1996.

[Pat00] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. *Des. Codes Cryptogr.*, 20(2):175–209, 2000.

[Pet13] Albrecht Petzoldt. *Selecting and Reducing Key Sizes for Multivariate Cryptography*. PhD thesis, Technische Universität, Darmstadt, July 2013.

[PR12] Keith Pardue and Benjamin Richert. Syzygies of semi-regular sequences (vol 53, pg 349, 2009). 56:1001–1003, 12 2012.

[Rei95] Miles Reid. *Undergraduate Commutative Algebra*. London Mathematical Society Student Texts. Cambridge University Press, 1995.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[Seg04] AJM Segers. Algebraic attacks from a Gröbner Basis perspective. Master's thesis, Technische Universiteit Eindhoven, 2004.

[Sha01] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, Cambridge, second edition, 2009.

[SK06] Makoto Sugita and Mitsuru Kawazoe. Relation between the XL algorithm and Grobner basis algorithms. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 89(1):11–18, 2006.

[SSS10] Rajesh P. Singh, Anupam Saikia, and B. K. Sarma. Little dragon two: An efficient multivariate public key cryptosystem. *CoRR*, abs/1005.5028, 2010.

[Ste06] Till Stegers. *Faugere's F5 Algorithm Revisited.* PhD thesis, Technische UniversitÂšat Darmstadt, 2006.

[Tho13] Enrico Thomae. *About the Security of Multivariate Quadratic Public Key Schemes.* PhD thesis, Ruhr Uiversität Bochum, 2013.

[Tra96] Carlo Traverso. Hilbert functions and the Buchberger algorithm. *J. Symbolic Comput.*, 22(4):355–376, 1996.

[Wol05] Christopher Wolf. *Multivariate Quadratic Polynomials in Public Key Cryptography.* PhD thesis, ESAT/COSIC, K.U.Leuven, 2005.

[Wol06] Christopher Wolf. Introduction to Multivariate Quadratic Public Key Systems and Their Applications. In *Proceedings of Yet Another Conference on Cryptography, YACC*, pages 44–55, 2006.

[WP] Christopher Wolf and Bart Preneel. Equivalent keys in multivariate quadratic public key systems.

[WP04] Christopher Wolf and Bart Preneel. Applications of Multivariate Quadratic Public Key Systems. 2004.

[YC04] Bo-Yin Yang and Jiun-Ming Chen. Theoretical analysis of XL over small fields. In *Information security and privacy. 9th Australasian conference, ACISP 2004, Sydney, Australia, July 13–15, 2004. Proceedings.*, pages 277–288. Berlin: Springer, 2004.

[YCYCY13] Jenny Yuan-Chun Yeh, Chen-Mou Cheng, and Bo-Yin Yang. Operating degrees for xl vs. f4/f5 for generic $\mathcal{M}q$ with number of equations linear in that of variables. 01 2013.