Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Criptografía Post-cuántica: Análisis de McEliece y una nueva versión con MPC

Autor: David Moreno Centeno

Tutor: Diego Ruano Benito

Agradecimientos

A los dos tutores de ambos trabajos fin de grado, Diego Ruano y Jose Ignacio Farrán por todo el tiempo empleado a lo largo de este año para guiarme y ayudarme a realizar el proyecto, y a mis padres, que me han dado el apoyo y ánimos necesarios a lo largo de la realización de todo el documento y también durante toda la carrera.

Resumen

La seguridad empleada en prácticamente todas comunicaciones realizadas actualmente utiliza una criptografía de clave pública o híbrida mediante el uso de sistemas criptográficos como el RSA, Gamal o de curva elíptica entre otros. Dichos sistemas aunque son actualmente seguros, en cuanto seamos capaces de construir un ordenador cuántico, se conoce un algoritmo que permite romperlos en tiempo polinómico. Por ello, actualmente se están estudiando distintos criptosistemas que sean resistentes a ataques realizados por un ordenador cuántico.

El presente documento se centra en el criptosistema de McEliece, el cual es uno de los criptosistemas resistente frente a estos ataques. En adicción se muestran los códigos Reed-Solomon, Goppa y de producto de matrices, que se pueden emplear, entre otros, para construir el criptosistema. Después vemos un posible ataque contra el criptosistema construido a partir de un código Reed-Solomon.

También, mostramos el método empleado por Gaborit para intentar eliminar la principal desventaja del criptosistema de McEliece preservando la seguridad, el gran tamaño de las claves. Por último se muestra un método innovador que nos permite reducir notablemente el tamaño de las claves del criptosistema mediante el empleo de códigos de producto de matrices.

Abstract

The security used in almost all communications currently performed a public key cryptography or hybrid through the use of cryptographic systems such as RSA, Gamal or elliptical curve among others. These systems although they are currently safe, as soon as we are able to build a quantum computer, it is known an algorithm that can break them in polynomial time. For this reason, different cryptosystems that are resistant to attacks carried out by a quantum computer are currently being studied.

This paper focuses on the McEliece cryptosystem, which is one of the cryptosystems resistant to these attacks. In addition, Reed-Solomon, Goppa and matrix product codes are shown, which can be used to build the cryptosystem. Afterwards, a possible attack against the cryptosystem built from a Reed-Solomon code is shown.

Also, it is shown the method used by Gaborit to try to eliminate the main disadvantage of the McEliece cryptosystem while preserving security,the large size of the keys. Finally, it shows an innovative method that allows us to significantly reduce the size of the cryptosystem keys by using matrix product codes.

Índice general

1. Introducción **1**
1.1. Motivación 5
1.2. Objetivos 5

2. Teoría de los códigos lineales **7**
2.1. Introducción 7
2.2. Descodificación 14
2.3. Código dual 18

3. Códigos RS y GRS **21**
3.1. Introducción 21
3.2. Matriz generatriz y de control 24
3.3. Codificación y descodificación 24

4. Códigos de Goppa **31**
4.1. Introducción 31
4.2. Descodificación 34
4.2.1. Algoritmo de Patterson 36

5. McEliece y Niederreiter **41**
5.1. Criptosistema de McEliece 41
5.1.1. Introducción 41
5.1.2. Descripción del criptosistema 42
5.1.3. Cifrado y descifrado del McEliece 42
5.1.4. Ventajas y desventajas del criptosistema 47
5.2. Criptosistema Niederreiter 48
5.2.1. Introducción 48
5.2.2. Descripción del criptosistema 48
5.2.3. Cifrado y descifrado del Niederreiter 49
5.3. Posibles ataques 50

6. Ataque contra códigos GRS **53**
6.1. Introducción 53
6.2. Ataque por filtración a códigos GRS 62
6.2.1. Ataque estándar 62
6.2.2. Ataque incompleto al criptosistema de McEliece 67
6.2.3. Ataque correcto al criptosistema de McEliece 68

7. Reducción de claves de Gaborit **75**
7.1. Introducción 75
7.2. Método de reducción de claves 78
7.2.1. Generación de claves 80
7.2.2. Cifrado 80
7.2.3. Descifrado 80
7.2.4. Tipo de códigos sugeridos 81
7.3. Criptoanálisis del método 81

8. Códigos de productos de matrices **85**
8.1. Introducción 85
8.2. Descodificación 95

9. McEliece con MPC **105**
9.1. Introducción 105
9.2. El nuevo criptosistema de McEliece 106
9.2.1. Generacion de claves 107
9.2.2. Cifrado 107
9.2.3. Descifrado 108

Bibliografía **113**

Capítulo 1

Introducción

Cuando una persona desea transmitir un mensaje a una o varias personas a través de un canal de comunicación, debe tener en cuenta que existe la posibilidad de que una persona ajena a la comunicación intente obtener dicho mensaje.

Para evitar este problema surge la criptografía, la cual se centra en el estudio de métodos que nos permitan convertir un mensaje de texto plano en una secuencia aleatoria de caracteres de un alfabeto para que dicho mensaje resulte ilegible para cualquier persona ajena a la comunicación que obtenga el mensaje.

Para ello, se puede emplear tanto la codificación como el cifrado, los cuales se centran en que la información sea transmitida de forma confidencial. Se pueden distinguir dos tipos de codificaciones en función de la longitud que tienen sus palabras; si todas las palabras de un código tienen la misma longitud se denomina código en bloque y si no lo son se denomina código de longitud variable.

Por tanto, aunque realizar la codificación o cifrado de un mensaje tiene como principal objetivo garantizar la seguridad de la comunicación establecida, también nos permite detectar y corregir cierta cantidad de errores que pueden surgir durante la transmisión del mensaje si empleamos cierto tipo de códigos en bloque o comprimir la información si en su lugar empleamos códigos de longitud variable, donde para ello tenemos que codificar los caracteres que se emplean de forma mas frecuente con las palabras del código de menor longitud.

Los primeros sistemas criptográficos, conocidos como clásicos o como cifrados de clave privada emplean una clave que debe conocer tanto el emisor como el receptor antes de comenzar la comunicación, ya que emplean dicha clave tanto para cifrar como para descifrar los distintos mensajes. Por ello, no eran suficientes para el intercambio de mensajes de forma segura mediante un canal inseguro ya que, dicha clave debe cambiarse cada cierto tiempo y para realizar esto de forma segura, es necesario reali-

zar una comunicación directa, para evitar que una persona ajena obtenga dicha clave privada.

En consecuencia, surgen los sistemas criptográficos de clave pública en 1976 gracias a Whitfield Diffie y Martin Hellman, los cuales, se basan en la teoría de la complejidad computacional. Estos sistemas criptográficos permiten que el descifrado sea imposible de forma práctica en un tiempo relativamente corto a no ser que el receptor conozca información que simplifique dicho proceso de descifrado, para ello se emplean funciones matemáticas cuyo cálculo es sencillo pero el cálculo de la función inversa es muy complejo. Dichas funciones son conocidas como funciones trampa o funciones de una vía. El principal ejemplo de función trampa es la función que se encarga de multiplicar dos números primos. Dicha operación se realiza rápidamente, sin embargo realizar la función inversa, es decir, obtener los dos números primos que multiplicados nos proporcionan un numero dado tiene una complejidad exponencial.

En los sistemas de clave pública cada persona de la comunicación tiene un par de claves:

- La clave pública y por tanto, conocida por cualquier persona que lo desee, la cual deberán emplear para cifrar las personas que deseen enviar un mensaje a la persona a la que pertenece dicha clave pública.
- La clave privada, solo conocida por la persona a quien pertenece, la cual permite descifrar los mensajes que hayan sido cifrados con la clave pública que le corresponde.

Estos sistemas de clave pública deben cumplir las llamadas condiciones de Diffie-Hellman, las cuales se corresponden con que el cálculo de las claves tanto pública como privada, así como el proceso de cifrado tiene que ser computacionalmente sencillo, mientras que el proceso de descifrado deberá ser computacionalmente imposible si no se conoce la clave privada. Ademas, debe ser computacionalmente imposible obtener la clave privada a partir de la pública y también dichos sistemas deben ser resistentes a ataques a texto claro elegido, es decir, ataques en los que se puede obtener texto cifrado correspondiente a un texto plano deseado y mediante un análisis de dicho texto poder descifrar cualquier otro mensaje cifrado, ya que la clave pública empleada para cifrar es conocida por cualquier persona.

Una desventaja de los cifrados asimétricos en comparación con los cifrados simétricos es que para tener una seguridad semejante es necesario emplear claves mas extensas. Debido a esto los cifrados asimétricos son mucho mas costosos computacionalmente y por ello, se suelen emplear los cifrados asimétricos dentro de una comunicación únicamente para el intercambio de una clave simétrica, que se genera y emplea

solo en esa comunicación, la cual permite una mayor rapidez en la transmisión de información manteniendo el nivel de seguridad. Este método se conoce como criptografía híbrida, debido al empleo de ambos tipos de cifrados.

Actualmente, se emplea la criptografía híbrida, para la cual se utilizan principalmente como sistemas criptográficos de clave pública el RSA (Rivest, Shamir y Adleman), el Gamal, el cual se basa en el problema matemático del logaritmo discreto, y el de curva elíptica. Anteriormente también se empleaba el criptosistema de Merkle-Hellman basado en el problema de la mochila aunque se dejó de emplear rápidamente debido a que fue criptoanalizado exitosamente por Shamir en 1982 [2].

Por tanto, todas las comunicaciones que realizamos actualmente dependen de la seguridad de dichos sistemas. Debido a esto, debemos hacernos una pregunta importante, ¿Son actualmente seguros estos sistemas criptográficos y lo seguirán siendo en un futuro próximo?

La respuesta es que actualmente son seguros. Sin embargo, el algoritmo cuántico de Shor, desarrollado por Peter Shor en el año 1998, es capaz de criptoanalizar exitosamente prácticamente todos los sistemas criptográficos de clave pública que se están empleando ya que dicho algoritmo permite descomponer un numero en sus factores primos en tiempo polinomial. Por tanto, si en un futuro próximo somos capaces de crear un ordenador cuántico suficientemente potente como para ejecutar el algoritmo, dichos sistemas de clave pública quedarían obsoletos. Por otro lado, remarcamos que los sistemas de clave privada son seguros frente a ataques que empleen el algoritmo de Shor.

Debido a esto, surge la necesidad de buscar otros sistemas de clave pública que sean capaces de resistir ataques realizados por un ordenador cuántico.

De momento se cree que los sistemas basados en Hash como por ejemplo el hash-tree de Merkle [3], los basados en teoría de códigos como por ejemplo el McEliece, o los basados en ecuaciones cuadráticas multivariantes como por ejemplo el de Patarin [4] son resistentes frente a ataques realizados tanto por ordenadores clásicos como por ordenadores cuánticos. Esto se debe a que en dichos sistemas no se ha encontrado de momento ninguna forma de aplicar el algoritmo de Shor.

Por tanto, deberíamos pensar porque no se están empleando ya estos criptosistemas si parecen ser resistentes frente a ataques realizados por ordenadores cuánticos. La respuesta se encuentra en que cada uno de ellos presenta algún tipo de desventaja que no presentan los sistemas de clave pública que se están empleando actualmente.

Por ejemplo, para el sistema de McEliece, que es el sistema basado en la teoría de códigos en que centraremos nuestro estudio en este trabajo fin de grado, su principal

inconveniente es que presenta un tamaño de claves mucho mas elevado que las del sistema RSA, lo cual incrementaría notablemente el tamaño que necesitaríamos emplear para guardar las distintas claves.

En este documento primero introducimos en el capítulo 2 unas nociones elementales de la teoría de códigos lineales necesarias para la correcta comprensión del documento. También mostramos algunos de los códigos lineales mas empleados en los capítulos 3 y 4, como son los códigos Reed-Solomon generalizados y los códigos de Goppa. Para ellos, se explica como obtener una matriz generatriz y de control y como realizar la codificación y descodificación de los mensajes que se deseen transmitir.

Tras explicar tanto el criptosistema de McEliece como el criptosistema de Niederreiter en el capítulo 5, para los cuales también mostramos como realizar el cifrado y descifrado de los mensajes así como de las ventajas y desventajas que presentan dichos criptosistemas, mostramos la inseguridad del criptosistema de McEliece implementado con códigos Reed-Solomon generalizados en el capítulo 6, mediante un ataque conocido como ataque de filtración. Para ello mostramos en primer lugar el ataque por filtración estándar, para el cual suponemos que conocemos dos códigos Reed-Solomon generalizados cuya dimensión dista en uno entre ambos códigos. Después, rebatimos un ataque de filtración incompleto contra el criptosistema de McEliece y por último mostramos un ataque de filtración exitoso sobre el criptosistema de McEliece, para el cual únicamente conocemos la matriz generatriz enmascarada de un código Reed-Solomon generalizado. De esta forma se busca mostrar que existen otros ataques efectivos frente a estos sistemas distintos del algoritmo de Shor dependiendo del tipo de códigos que empleemos para su implementación.

Por ultimo, explicamos otro tipo de códigos lineales en el capítulo 8, llamados códigos de producto de matrices, para los cuales vemos como obtener los parámetros fundamentales y un algoritmo que permite realizar la descodificación de dichos códigos cuando estos cumplen unas condiciones que no son muy restrictivas. Después mostramos un método innovador en el capítulo 9 que permite implementar el criptosistema de McEliece, eliminando su principal desventaja relacionada con el tamaño de las claves del criptosistema, mediante el empleo de los códigos de producto de matrices. La reducción de las claves del criptosistema se basa en una idea semejante a la realizada por Gaborit en el año 2004, la cual también mostramos en el capítulo 7 junto con una pequeña introducción a los códigos cíclicos y cuasi-cíclicos, necesarios para la correcta comprensión del procedimiento que se explica en dicho capítulo.

A lo largo de los diferentes capítulos, se han realizado una serie de ejemplos para facilitar la comprensión de toda la teoría mostrada. Para su realización, en muchos de ellos se ha utilizado el lenguaje de programación llamado SageMath, donde todo el código que se ha ido implementando para la realización de los ejemplos se muestra y

explica detalladamente en la parte del trabajo fin de grado de informática [1].

1.1. Motivación

Como ya hemos mencionado, en cuanto seamos capaces de construir ordenadores cuánticos, todos los métodos implementados actualmente que permiten realizar cualquier comunicación de forma segura quedarán obsoletos.

Debido a esto, se realiza en este trabajo fin de grado un estudio del criptosistema de McEliece, al ser un método alternativo a los empleados actualmente, resistente frente al algoritmo de Shor, el cual permite paliar la situación crítica en la que nos encontraríamos si no fuéramos capaces de realizar ninguna comunicación segura.

El principal aspecto que ha proporcionado el interés para la realización de este documento es la gran relevancia que presenta la seguridad en cualquier tema que se desee realizar, junto con la oportunidad de poder ampliar mis conocimientos sobre dicho aspecto, realizando un estudio de la teoría de códigos, los cuales hacen referencia a una serie de conceptos completamente nuevos para el alumno.

1.2. Objetivos

Ahora mostramos una serie de objetivos que se pretenden alcanzar tras la realización del presente trabajo fin de grado.

- Estudio de la teoría básica de códigos lineales.
- Conocimiento de una serie de códigos lineales como son los códigos Reed-Solomon y Reed-Solomon generalizados, códigos cíclicos y cuasi-cíclicos, códigos de Goppa y códigos de producto de matrices, mostrando para todos ellos, a excepción de los códigos cíclicos y cuasi-cíclicos, un algoritmo capaz de corregir una cierta cantidad de errores.
- Entender el criptosistema de McEliece y de Niederreiter, sus ventajas y desventajas y como realizar tanto el cifrado como el descifrado.
- Analizar la seguridad del criptosistema de McEliece dependiendo del tipo de códigos con el que se implementan. Mostrar un criptoanalisis exitoso del McEliece con códigos Reed-Solomon generalizados, conocido como ataque por filtracion.
- Mostrar uno de los primeros intentos, realizado por Gaborit, para reducir el tamaño de las claves del criptosistema de McEliece.

- Explicar un método innovador que permite implementar el criptosistema de McEliece reduciendo de forma notable el tamaño de las claves que es necesario emplear.
- Aprender los conocimientos básicos del lenguaje de programación Python necesarios para entender el funcionamiento detallado del lenguaje de programación SageMath, el cual nos permite realizar distintas operaciones matemáticas, las cuales resultarían costosas de realizar de forma escrita.

Capítulo 2

Teoría de los códigos lineales

2.1. Introducción

Los códigos lineales son códigos en bloque que poseen una estructura algebraica, la cual permite que la codificación y descodificación realizada sea menos costosa computacionalmente. Por ello, todos los códigos que emplearemos en adelante serán lineales.

Definición 2.1. Un código lineal C de dimensión k, es un k-subespacio vectorial de un espacio vectorial $\mathbb{F}_q^n$. Normalmente, los códigos lineales se suelen denotar con los llamados parámetros fundamentales del código, $[n, k]$, donde, n indica la longitud de los elementos del subespacio vectorial, k indica la dimension del subespacio vectorial, es decir, del código lineal.

Definición 2.2. Una palabra del código es un elemento del código lineal C, y todas las palabras de código tienen la misma longitud.

Nota 2.3. La cantidad de palabras de un código C, es q^k, donde q hace referencia al cuerpo finito $\mathbb{F}_q$ de q elementos con el que estamos trabajando.

Ejemplo 2.1. Vamos a emplear un código lineal, que denotaremos por C, el cual tiene como parámetros fundamentales $n = 7$ y $k = 4$ en $\mathbb{F}_2$, el cual es un cuerpo finito formado únicamente por los elementos 0 y 1, por ello, también se le llama cuerpo binario.

Obviamente, según las definiciones vistas, las palabras del código lineal $[7, 4]$ serán vectores de longitud 7, donde cada una de sus coordenadas se corresponden con elementos de $\mathbb{F}_2$. La cantidad total de palabras de código es $2^4 = 16$.

De hecho, emplearemos *Sagemath* para la construcción de un código lineal aleatorio de parámetros $[7, 4]$:

```
In: C= codes.random_linear_code(GF(2),7,4);
```

Definición 2.4. Una matriz generatriz G de un código lineal $[n,k]$, es una matriz $k \times n$ cuyas filas son vectores linealmente independientes del código lineal que además lo generan, es decir, dichas filas forman una base del código lineal como espacio vectorial.

Nota 2.5. Podemos observar de la definición 2.4 que un código lineal C, al igual que puede tener distintas bases, también puede tener distintas matrices generatrices, aunque todas ellas son semejantes, y por tanto, existirá una matriz invertible que nos permita obtener una matriz generatriz a partir de la otra. Aun así, hay que destacar que distintas matrices generatrices de un código generan distintas codificaciones, ya que $C = \{aG \mid a \in \mathbb{F}_q^k\}$.

Nota 2.6. También podemos relacionar a cada matriz generatriz G de un código C con una aplicación lineal inyectiva, f, que llamaremos aplicación de codificación:

$$f : \mathbb{F}_q^k \longrightarrow C \subset F_q^n$$

Definición 2.7. Dados dos códigos lineales C_1 y C_2 de igual longitud, n, sobre $\mathbb{F}_q$, decimos que son códigos equivalentes, si existe una permutación π del conjunto de indices $\{1, \cdots, n\}$ de forma que $C_2 = \{\pi(c) \mid c \in C_1\}$

Definición 2.8. Dado un código lineal C, diremos que es un código sistemático si y solo si su matriz generatriz G es de la forma $G = (I_k|A)$, siendo I_k la matriz identidad de tamaño $k \times k$.

Nota 2.9. La matriz generatriz de un código sistemático se llama matriz generatriz en forma estándar.

Cuando el código es sistemático, al realizar la codificación de un vector se obtiene como resultado una palabra del código formada por el vector sin codificar seguido, hasta completar la longitud de la palabra del código, de coordenadas llamadas de control.

Las palabras de un código lineal $[n,k]$ en forma sistemática tienen k símbolos de información y $n-k$ símbolos redundantes, los cuales permiten validar que los k símbolos de información obtenidos son correctos y por tanto, no tienen errores.

Esto facilita la obtención de la información de forma correcta a pesar de que dicha información, bien durante su transmisión a través del canal de comunicación o cualquier otra circunstancia, haya sufrido una cierta cantidad de errores.

Proposición 2.10. *Dado un código lineal, C, de parámetros $[n,k]$, es equivalente a un único código sistemático.*

Demostración. Sea G una matriz generatriz del código de tamaño $k \times n$. Entonces G posee k columnas linealmente independientes. Realizando una permutación de forma que tengamos las k primeras columnas linealmente independientes obtenemos un código equivalente que tiene como matriz generatriz $G' = (A|B)$, siendo A una matriz regular. Ahora aplicando el método de Gauss podemos transformar la matriz A en la matriz identidad I_k y por tanto, obtenemos un código sistemático como queríamos.

Ademas, como la forma escalonada reducida de una matriz es única y las matrices generatrices del código C son equivalentes, entonces tenemos que el código sistemático debe ser único. □

Definición 2.11. Una matriz de control, H, de un código lineal $[n, k]$, C, es una matriz $(n-k) \times n$ para la cual se cumple que $c \in C \Leftrightarrow Hc^t = 0, \forall c \in \mathbb{F}_q^n$.

Debido a ello, es claro que si G es la matriz generatriz y H es la matriz de control de un código lineal C entonces tenemos que $GH^t = HG^t = 0$

Ejemplo 2.2. Continuando con el ejemplo 2.1, utilizando la sentencia:

```
In: G=C.generator_matrix()
```

Sagemath nos proporciona una matriz generatriz de dicho código, cuyas filas hacen referencia a una base del código:

$$G = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

La aplicación lineal asociada a esta matriz generatriz viene definida de la siguiente forma:

$$f : \mathbb{F}_2^4 \longrightarrow C \subset \mathbb{F}_2^7$$

$$a = (a_1, a_2, a_3, a_4) \longmapsto c = (a_3 + a_4, a_3, a_1 + a_2, a_2 + a_3 + a_4, a_2 + a_3, a_1 + a_4, a_1 + a_2 + a_4)$$

Según lo que hemos visto en la proposición 2.10, existe un código equivalente al de partida cuya matriz generatriz se encuentra en forma estándar, para su obtención empleamos el siguiente código en *Sagemath*:

```
In: G'=C.systematic_generator_matrix()
```

y obtenemos la matriz, G', correspondiente a la matriz generatriz en forma estándar:

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

La aplicación lineal asociada a esta matriz generatriz en forma estándar viene definida de la siguiente forma:

$$f : \mathbb{F}_2^4 \longrightarrow C \subset \mathbb{F}_2^7$$

$$a = (a_1, a_2, a_3, a_4) \longmapsto c = (a_1, a_2, a_3, a_4, a_1 + a_2 + a_4, a_2 + a_3 + a_4, a_1 + a_2 + a_3)$$

Esto es debido a que cualquier palabra de código, c, se forma multiplicando el bloque de mensaje por la matriz generatriz: $aG = c \in C$.

En adicción, podemos observar que el código generado a partir de la matriz G' se corresponde con el código de Hamming $[7, 4]$.

Dichos códigos fueron creados por Richard Hamming en el siglo XX, los cuales se caracterizan por ser de los primeros en la teoría de codificación y por ser capaces de detectar y corregir un único error pero no de detectar ni siquiera dos errores.

En cuanto al cálculo de la matriz de control según lo visto en la definición 2.11, calculamos la matriz tal que tras multiplicar la matriz generatriz por la matriz calculada obtengamos la matriz nula y entonces la matriz de control se corresponde con la traspuesta de la matriz calculada.

El código empleado para ello es:

```
In: H=G.right_kernel_matrix();
```

y la matriz de control, H, obtenida para el código C es:

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Vimos que los códigos lineales eran capaces de corregir cierta cantidad de errores, pero para saber la cantidad de errores que es capaz un código lineal de corregir es necesario introducir los conceptos de peso y distancia mínima del código. Dichos conceptos también fueron descubiertos por Richard Hamming, por lo que son conocidos añadiendo su nombre.

Definición 2.12. El peso de Hamming de un vector $c = (c_1, ..., c_n) \in \mathbb{F}_q^n$, denotado por $\omega(c)$, es el numero de coordenadas no nulas del vector, es decir, $\omega(c) = \#\{i \mid c_i \neq 0,\ 1 \leq i \leq n\}$.

Nota 2.13. Una vez definido el peso de Hamming de un vector, podemos deducir que el peso mínimo de un código lineal C sera el mínimo de los pesos de Hamming de las palabras del código, es decir,

$$\omega(C) = min\{\omega(c) \mid c \in C \setminus \{0\}\}$$

Definición 2.14. La distancia de Hamming entre dos vectores $a, b \in \mathbb{F}_q^n$, denotado por $d(a, b)$, es el número de coordenadas donde ambos vectores no coinciden, es decir, $d(a, b) = \#\{i \mid a_i \neq b_i,\ 1 \leq i \leq n\}$.

Nota 2.15. Es fácil comprobar que la distancia de Hamming es una distancia y por tanto el espacio vectorial $\mathbb{F}_q^n$ es un espacio métrico con dicha distancia. También podemos observar que $\omega(c) = d(0, c)\ \forall c \in \mathbb{F}_q^n$.

De igual forma que ocurría con el peso, la distancia mínima de un código lineal C sera la mínima distancia de Hamming entre cualquier par de palabras del código distintas. Además, para los códigos lineales, existe una relación entre la distancia mínima y peso mínimo de un código:

Proposición 2.16. *Dado un código lineal $[n, k]$, C, la distancia mínima de C es igual a el peso mínimo de C.*

Demostración. Sea $c \in C$, una palabra de código de peso mínimo. Entonces tenemos que $\omega(c) = d(0, c)$, y como $0 \in C$ por ser un subespacio vectorial, deducimos que $d(C) \leq \omega(C)$.

Por otro lado, sean $a, b \in C$, palabras del código de distancia mínima. Tenemos que $d(a, b) = d(a - b, 0) = d(0, a - b) = \omega(a - b)$ y como $a - b \in C$ debido a que C es un subespacio vectorial, entonces tenemos que $\omega(C) \leq d(C)$ y por tanto la igualdad es cierta. □

Según la definición de distancia mínima de un código que hemos dado, tenemos que comparar cada par de palabras del código o el peso de cada palabra del código. Esto en general es difícil de realizar, ya que cuando tengamos códigos con gran cantidad de palabras hay que realizar multitud de comprobaciones. A pesar de ello, se puede emplear la matriz de control del código, la cual nos facilita el trabajo como podemos observar a partir de la siguiente proposición:

Proposición 2.17. *Dado un código lineal, C, con matriz de control, H, y distancia mínima d. Si r es un numero entero positivo, tenemos que $d > r \Leftrightarrow$ cualquier combinación de r columnas de la matriz de control H son linealmente independientes.*

Demostración.

$\Rightarrow$) Para ello veamos el contrarreciproco: Si existen r columnas linealmente dependientes en la matriz de control, entonces existirá un vector c, con coordenadas que proporcionan dicha combinación lineal y por tanto, tenemos que $Hc^t = 0$, es decir, $c \in C$, y como $\omega(c) \leq r$ entonces $d \leq r$

$\Leftarrow$) Ahora si partimos de que cualquier combinación de r columnas de H son independientes, entonces ningún vector c puede tener coordenadas que generen una combinación lineal a partir de solamente esas columnas. Por tanto, para ese caso tendríamos que $Hc^t \neq 0$ y entonces $c \notin C$. Debido a esto, si $c \in C$, necesariamente $\omega(c) > r$ como queríamos.

□

Corolario 2.18. *Dado un código lineal C, con matriz de control H. La mínima distancia de C, $d(C)$, es igual al mínimo numero de columnas igualmente dependientes de H.*

La proposición anterior también nos proporciona una cota superior para la distancia mínima de cualquier código lineal, conocida como cota de Singleton:

Corolario 2.19. ***(cota de Singleton)*** *Dado un código lineal $[n, k]$, C, cuya mínima distancia es d, se cumple que $d \leq n - k + 1$*

Demostración. Sea H la matriz de control de C. Entonces el rango de H es $n - k$ y por tanto $n - k + 1$ columnas de H son linealmente dependientes y por la proposición anterior obtenemos que $d \leq n - k + 1$. □

Los códigos que satisfacen la igualdad en la cota de Singleton, es decir, que cumplen $d = n - k + 1$, son conocidos como códigos de máxima distancia separable o MDS.

Nota 2.20. A partir de los conceptos de distancia y peso mínimo definidos, los cuales son válidos para cualquier tipo de código, podemos observar que estos son los que nos permiten detectar cierta cantidad de errores, siendo e un vector, llamado vector de errores, con longitud igual a la longitud de las palabras del código que se esta empleando y con peso $\omega(e) < d$, siendo d la distancia mínima de dicho código.

En esta situación, si obtenemos una palabra con menos de d errores, esta no pertenecerá al código que estamos empleando y por tanto podremos percatarnos que en esa palabra hay errores. Además, si dicho error tiene peso suficientemente pequeño como para que exista una única palabra del código con menor distancia de Hamming a la palabra obtenida, seremos capaces de identificar el error, siendo la única palabra con menor distancia de Hamming la palabra sin errores.

Por otro lado, si el error tiene peso $\omega(e) \geq d$, entonces una palabra de código, c, con dicho error, puede dar lugar a otra palabra del código, $c' = c + e$, y por tanto provoca que en ocasiones no nos percatemos del error producido.

Debido a esto, una vez detectada una palabra con cierta cantidad de errores, debemos preguntarnos cuantos errores somos capaces de corregir. Para ello veamos ahora la capacidad correctora de dichos códigos:

Definición 2.21. Dado un código lineal, C, diremos que es corrector de t errores si para cualesquiera dos palabras del código, $a, b \in C$ y para cualesquiera dos vectores de errores,e y e' con $\omega(e), \omega(e') \leq t$, tenemos que $a + e \neq b + e'$.

El siguiente teorema nos proporciona una forma mas sencilla de obtener la capacidad correctora del código:

Teorema 2.22. *Un código $[n, k]$ es corrector de t errores $\Leftrightarrow t < \frac{d}{2}$, siendo d la distancia mínima de dicho código.*

Demostración.

$\Rightarrow$) Realizaremos una reducción al absurdo: Supongamos que $t \geq \frac{d}{2}$. Tomamos una palabra del código, c, tal que $\omega(c) = d$. Ahora cambiamos $[\frac{d}{2}]$ posiciones no nulas a nulas en c obteniendo y. Se cumple entonces que $\omega(y) = d(0, y) \leq d - [\frac{d}{2}] \leq t$ y $\omega(y - c) = d(c, y) \leq [\frac{d}{2}] \leq t$. Como el código es corrector de t errores entonces se cumple por definición que para cualquier par de palabras de código distintas, a los que introducimos errores de peso menor que t deben seguir siendo distintas. Por ello, llegamos a contradicción, ya que 0 y c pertenecen al código y se cumple que $0 + y = c + (y - c)$.

$\Leftarrow$) Realizamos de nuevo otra reducción al absurdo: Supongamos que $t < \frac{d}{2}$ y que el código no es corrector de t errores, es decir, existen dos palabras de código, a y b, con errores, e_a y e_b, ambos de peso menor que t, tal que $a + e_a = b + e_b$. Entonces se cumple que $a - b = e_b - e_a$ y por tanto $\omega(a - b) = \omega(e_b - e_a) \leq 2t < d$. Por ello, llegamos a una contradicción al ser $a - b$ otra palabra de código cuyo peso es inferior al peso mínimo del código.

□

Ejemplo 2.3. Siguiendo con el ejemplo 2.2, *Sagemath* tiene ya definida la función *hamming_weight*(), que proporciona el peso de Hamming de una palabra de código determinada y *minimum_distance*(), que proporciona la mínima distancia del código. Empleamos dicho código:

```
In: C.minimum_distance()

Out: 3
```

Por tanto obtenemos que el peso mínimo del código es, $\omega(C) = 3$, el cual también podíamos obtener observando en la matriz de control que a partir de la suma de la cuarta y quinta columnas obtenemos la sexta.

Por la cota de Singleton obtenemos que $3 \leq 7 - 4 + 1 = 4$ y por tanto no es un código MDS.

Además, es claro que nuestro código C es capaz de corregir un único error, ya que por el teorema 2.22, los t que lo cumplen son 0 y 1.

Una vez que sabemos la distancia mínima y capacidad correctora de un código lineal, vamos a ver una forma de corregir dichos errores mas eficiente que la introducida en la nota 2.20, lo cual se debe a la estructura algebraica que presentan los códigos lineales.

2.2. Descodificación

En primer lugar vamos a introducir el concepto de síndrome de un código lineal, el cual, proporcionará información esencial sobre el error producido en una palabra de código transmitida.

Definición 2.23. Dado un código lineal $[n, k]$, C, con matriz de control H e $y \in \mathbb{F}_q^n$. Diremos que el síndrome de y, denotado por $s(y)$, viene dado por el vector obtenido de la multiplicación de la matriz de control con y, es decir,

$$s(y) = Hy^t$$

De esta definición, vemos que $s(y) \in \mathbb{F}_q^{n-k}$. También, que el síndrome de y será el vector nulo si y solo si $y \in C$. Por ello, las palabras de código no aportan ninguna información al síndrome, pero si definimos el síndrome como la siguiente aplicación:

$$s : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^{n-k}$$

$$y \longmapsto Hy^t$$

Podemos observar que es una aplicación lineal al estar definido como el producto de una matriz por un vector. Por ello, si tenemos una palabra del código, $c \in C$, con

una cantidad de errores, donde e es el vector de errores, la palabra resultante, $y = c + e$, tiene el mismo síndrome que el vector de errores ya que $s(y) = Hy^t = s(c + e) = H(c + e)^t = Hc^t + He^t = He^t = s(e)$. Esto nos facilita la corrección de errores en el proceso de descodificación como podemos ver a continuación.

Definición 2.24. Dado un código lineal $[n, k]$, C, y un elemento $a \in \mathbb{F}_q^n$. Llamamos cogrupo que contiene a a, al conjunto dado por:

$$a + C = \{a + c \mid c \in C\}$$

Proposición 2.25. *Dos elementos de $\mathbb{F}_q^n$ están en el mismo cogrupo si y solo si tienen el mismo síndrome.*

Demostración.

$\Rightarrow$) Sean x, y pertenecientes al mismo cogrupo $(a + C)$. Entonces $x = a + c$ con $c \in C$ y $y = a + c'$ con $c' \in C$. Por tanto ambas palabras tendrán el mismo síndrome ya que si H es la matriz de control del código, tenemos que $s(x) = Hx^t = H(a + c)^t = Ha^t = H(a + c')^t = s(y)$

$\Leftarrow$) Si x, y tienen el mismo síndrome se cumple que $Hx^t = Hy^t$ y entonces $H(x - y)^t = 0$, lo cual se cumple si y solo si $x - y \in C$. Por tanto, como $x = x + 0$ e $y = x + (y - x)$, se cumple que ambas están en el mismo cogrupo.

□

Definición 2.26. Llamaremos líder de un cogrupo al elemento de menor peso dentro de dicho subconjunto, el cual debe ser único. Por tanto, es posible que algunos cogrupos no posean líder.

A partir de esto, tenemos un método para la corrección de errores:

Si partimos de una palabra del código, c, la cual ha sufrido cierta cantidad de errores convirtiéndose en el vector $y = c + e$, es claro que tanto y como e pertenecen al mismo cogrupo y por ello tienen el mismo síndrome. Entonces calculamos el síndrome de y y si posee un líder, se podrá realizar la descodificación restando a y el líder del cogrupo, que será el error cometido. Si no posee líder no se podrá realizar la descodificación.

Proposición 2.27. *Cada cogrupo tiene como máximo un elemento de peso menor o igual que t.*

Demostración. Realizamos una reducción al absurdo, supongamos que existen x, y, pertenecientes al mismo cogrupo, distintas, no nulas y con $\omega(x), \omega(y) \leq t$. Entonces, sabemos que $x - y \in C$ y como $\omega(x - y) \leq 2t$, tendríamos que existe una palabra

del código no nula cuyo peso es inferior a la distancia mínima del código, lo cual es absurdo.

□

Proposición 2.28. *Dado un código lineal $[n,k]$, diremos que corrige t errores si y solo si todas las palabras de peso menor o igual que t son lideres de subconjuntos.*

La demostración de la proposición 2.28 se puede encontrar en [5].

Ejemplo 2.4. Siguiendo con el ejemplo 2.3, vamos a emplear el código lineal $[7,4]$ obtenido en forma sistemática, el cual, vimos que permite la corrección de un error.

Ahora vamos a obtener las $2^4 = 16$ posibles palabras del código usando la siguiente función de *Sagemath*:

```
In: C.list()
```

Por tanto, el código C esta constituido por:

$C = \{(0,0,0,0,0,0,0), (1,0,0,0,0,1,1), (0,1,0,0,1,0,1), (1,1,0,0,1,1,0),$
$(0,0,1,0,1,1,0), (1,0,1,0,1,0,1), (0,1,1,0,0,1,1), (1,1,1,0,0,0,0),$
$(0,0,0,1,1,1,1), (1,0,0,1,1,0,0), (0,1,0,1,0,1,0), (1,1,0,1,0,0,1),$
$(0,0,1,1,0,0,1), (1,0,1,1,0,1,0), (0,1,1,1,1,0,0), (1,1,1,1,1,1,1)\}$

Como el síndrome pertenece a el cuerpo finito $\mathbb{F}_2^3$, hay un total de $2^3 = 8$ síndromes. Emplearemos una función generada en *Sagemath* que nos proporcione, si existe, el líder de cada síndrome, para ello emplearemos la función *syndrome*() ya implementada en los códigos lineales en *Sagemath*, la cual proporciona el síndrome de un vector dado como parámetro:

```
In: def lidersindrome(C):
        q=C.base_ring().cardinality();
        n=C.random_element().length();
        k=C.dimension();
        j=q^(n-k); #tamaño de la lista de los posibles sindromes
        lider=[(n+1) for i in [1..j]];
        peso=[("") for i in [1..j]];
        for i in VectorSpace(IntegerModRing(q),n).list():
            s=C.syndrome(i);
            posicion=0;
            for j in range(n-k):
                posicion=posicion+ZZ(s[j])*q**(n-k-1-j);
            if  peso[posicion]>i.hamming_weight():
                lider[posicion]=i;
```

```
            peso[posicion]=i.hamming_weight();
        elif peso[posicion]==i.hamming_weight():
            lider[posicion]="null";
    for h in lider:
        print ('sindrome: ', C.syndrome(h), 'lider: ', h);
```

Para nuestro ejemplo podemos observar en la tabla 2.1 los siguientes lideres para cada posible síndrome obtenidos a partir de la función *lidersindrome*() creada.

Síndrome	Líder
$(0,0,0)$	$(0,0,0,0,0,0,0)$
$(0,0,1)$	$(0,0,0,1,0,0,0)$
$(0,1,0)$	$(0,1,0,0,0,0,0)$
$(0,1,1)$	$(0,0,0,0,0,1,0)$
$(1,0,0)$	$(1,0,0,0,0,0,0)$
$(1,0,1)$	$(0,0,0,0,1,0,0)$
$(1,1,0)$	$(0,0,1,0,0,0,0)$
$(1,1,1)$	$(0,0,0,0,0,0,1)$

Tabla 2.1: Lideres y sindromes.

Si introducimos ahora un error en una de las palabras del código C, podemos corregir dicho error según lo visto en la proposición 2.28 ya que el código tiene una capacidad correctora de $t = 1$. Por ejemplo, tomamos la palabra $c = (1,0,1,1,0,1,0)$ y añadimos un error en la primera coordenada obteniendo $y = (0,0,1,1,0,1,0)$. Ahora obtenemos el síndrome de y, el cual es, $s(y) = (1,0,0)$.

Mirando ahora la tabla 2.1 tenemos que el líder de ese síndrome es $(1,0,0,0,0,0,0)$. el cual coincide con el error introducido.

A partir de este método es posible realizar la descodificación de cualquier código lineal. Ademas es mas eficiente que la descodificación de códigos vista en la nota 2.20, ya que en lugar de tener que comparar con todas las palabras simplemente hay que buscar el líder del síndrome correspondiente. Sin embargo, podemos observar que este método sigue siendo poco eficiente ya que en cuanto trabajemos con códigos de dimensión y longitud extensa, como tenemos que almacenar un total de q^{n-k} síndromes y lideres, es necesario emplear una gran cantidad de memoria para su almacenamiento.

Es por esto, que para los códigos que veremos mas adelante, para realizar su descodificación, primero localizaremos las posiciones donde se encuentran los errores y

después, en caso de que el código no sea binario, se calcula el valor de dichos errores. Entonces, realizando la resta obtenemos la palabra del código sin errores.

2.3. Código dual

Si partimos de un código lineal, C, de parámetros $[n, k]$, entonces su matriz de control, H, al tener rango máximo, puede coincidir con la matriz generatriz de otro código lineal C', donde ambos códigos están definidos sobre $\mathbb{F}_q$. Podemos observar que al tener C dimension k, necesariamente C' tiene que tener dimension $n - k$ y una matriz de control de C' coincidirá con una matriz generatriz de C, G, ya que por la definición 2.11 sabemos que $GH^T = 0$.

Definición 2.29. Llamaremos código dual de un código lineal, C, y lo denotaremos por $C^\perp$, al código C'.

Si consideramos como producto interno sobre $\mathbb{F}_q^n$, denotado por $\langle \cdot \rangle$

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \in \mathbb{F}_q \text{ con } x, y \in \mathbb{F}_q^n$$

se cumple la siguiente proposición:

Proposición 2.30. *Dado un código lineal C, se cumple que el código dual, $C^\perp$, es el ortogonal de C respecto del producto interno anterior.*

Corolario 2.31. *Dado un código lineal C, se cumple que el dual del código dual coincide con el código lineal de partida, es decir, $(C^\perp)^\perp = C$.*

Ejemplo 2.5. Continuando con el ejemplo 2.4, vamos a calcular el código dual del código C. Para ello, se emplea el siguiente código de *Sagemath*:

```
In: dual= C1.dual_code();
    dual

Out: [7, 3] linear code over GF(2)
```

Podemos observar que se cumple que el código dual tiene dimension $7 - 4 = 3$ y además, todas las palabras del código dual son ortogonales respecto del producto interno de cualquier palabra de C, ya que por construcción, su producto con la matriz generatriz de C es el vector nulo:

```
In: G*dual.random_element()
```

```
Out: (0, 0, 0, 0)
```

Proposición 2.32. *Dado un código lineal, C, si dicho código es MDS se cumple que $C^{\perp}$ es también un código MDS.*

Demostración. Consideramos la matriz generatriz del código C, G. Como C es MDS, tenemos que cualesquiera k columnas son linealmente independientes en la matriz G. Como la matriz generatriz de C coincide con la matriz de control de $C^{\perp}$ entonces se cumple que $d > k$. Por otro lado, como la dimension del código dual, $C^{\perp}$, es $n - k$, por la cota de Singleton tenemos que $d \leq n - (n - k) + 1$, y entonces $d \leq k + 1$. Luego tenemos la igualdad y por ello el código dual es un código MDS.

□

Definición 2.33. Un código lineal, C, es autodual si su código dual es el mismo código, es decir, si $C = C^{\perp}$.

Nota 2.34. Podemos observar, que una condición necesaria para que un código, C, sea autodual es que su longitud debe ser par, ya que tiene que ocurrir que $k = n - k \Leftrightarrow n = 2k$.

Capítulo 3

Códigos Reed-Solomon

3.1. Introducción de los códigos Reed-Solomon y Reed-Solomon generalizados

Los códigos Reed-Solomon y códigos Reed-Solomon generalizados, también conocidos de forma abreviada como códigos RS y códigos GRS respectivamente, son un tipo de códigos creados por Irving S. Reed y Gustave Solomon en el año 1960. Dichos códigos pertenecen a la categoría Forward Error Correction (FEC).

Estos códigos resultan ser muy útiles para la corrección de errores a ráfagas que afectan a bloques contiguos. Por ello, se emplean en una gran diversidad de casos, por ejemplo, en la tecnología DSL y variantes como ADSL y VDSL, en distintas unidades de almacenamiento como discos duros, CD, DVD, BlueRay y también en códigos de barras y radiodifusión digital actual como DVB y sus variantes.

Definición 3.1. Un código Reed-Solomon es un código sobre un cuerpo finito $\mathbb{F}_q$, con un vector $a = (a_1, ..., a_n) \in \mathbb{F}_q^n$, donde $a_i \neq a_j \, \forall i \neq j$, y n es un número entero con $0 < n \leq q - 1$. Normalmente, se considera $n = q - 1$ para tener la longitud máxima que puede alcanzar el código, y de esta forma es además un código cíclico.

Sea $L_k \subset \mathbb{F}_q[x]$ el espacio vectorial formado por todos los polinomios de grado menor que k, donde k es un entero con $0 < k \leq n$. Es evidente que la dimensión de L_k es k debido a que cualquier polinomio de grado menor que k se puede generar como combinación lineal de los elementos $\{1, x, ..., x^{k-1}\}$ sobre el cuerpo $\mathbb{F}_q$, y al ser dichos elementos linealmente independientes, forman una base de L_k. Entonces el código Reed-Solomon asociado al vector a, se denota por $RS_k(a)$, y esta formado por los vectores obtenidos al evaluar los polinomios de L_k en las coordenadas del vector a, es decir, $RS_k(a) = \{(f(a_1), ..., f(a_n)) \mid f \in L_k\}$.

Nota 3.2. Podemos observar que al ser $a \in \mathbb{F}_q^n$, entonces todas las manipulaciones rea-

lizadas mediante el código necesitan únicamente de operaciones en el propio cuerpo.

Las principales características de los códigos Reed-Solomon son las enunciadas en la siguiente proposición:

Proposición 3.3. *Dado un código de Reed-Solomon, $RS_k(a)$ se tiene que:*

a) $RS_k(a)$ es un código lineal.

b) $RS_k(a)$ es un código MDS (código de máxima distancia separable).

Demostración.

a) Sean $m_1, m_2 \in \mathbb{F}_q^k$ dos mensajes cualesquiera, cuyos polinomios correspondientes son $f(x), g(x) \in L_k$ respectivamente.

Sean $c_1, c_2 \in \mathbb{F}_q^n$ la codificación de m_1 y m_2 respectivamente, es decir, $c_1 = (f(a_1), ..., f(a_n))$ y $c_2 = (g(a_1), ..., g(a_n))$.

Sea $b \in \mathbb{F}_q$, es evidente que $bf(x)$ y $f(x) + g(x)$ son polinomios pertenecientes al espacio vectorial L_k.

Por ello, si denotamos $h(x) = f(x) + g(x)$, polinomio correspondiente del mensaje $m_3 = m_1 + m_2$, tenemos que $c_3 = (h(a_1), ..., h(a_n)) = (f(a_1), ..., f(a_n)) + (g(a_1), ..., g(a_n)) = c_1 + c_2$

De igual forma, si denotamos $h(x) = bf(x)$, polinomio correspondiente del mensaje $m_3 = bm_1$, tenemos que $c_3 = (h(a_1), ..., h(a_n)) = (bf(a_1), ..., bf(a_n)) = b(f(a_1), ..., f(a_n)) = bc_1$

Por tanto, el código $RS_k(a)$ es un código lineal.

b) Debido a que el código $RS_k(a)$ es lineal, tenemos que la distancia, d, entre una palabra fija del código, c_0, y cualquier otra palabra del código distinta, c, es independiente de c_0, al ser $c - c_0$ otra palabra del código. Por ello, tenemos que $d = min\{d(c, c_0) \mid c, c_0 \in RS_k(a)\} = min\{d(c', 0) \mid c' \in RS_k(a)\}$, siendo $c' = c - c_0$.

Por tanto, tomamos $m_1 \in \mathbb{F}_q^k$ un mensaje cuyo polinomio correspondiente es $f(x) \in L_k$. Es claro que f es un polinomio no nulo de grado a lo sumo $k - 1$. Entonces f tiene a lo sumo $k - 1$ raíces. Si denotamos por $c_1 = (f(a_1), ..., f(a_n))$, la codificación de m_1, tenemos que $d = \{d(c_1, 0)\} = \{\omega(c_1)\}$. Como su peso será n menos el numero de ceros, el cual coincide con n menos el numero de raíces del polinomio f (al ser inyectiva la evaluación de los polinomios por ser $k \leq n$), entonces su peso es al menos $n - (k - 1)$, y por ello $d \geq n - k + 1$.

Por otra parte, como la cota de Singleton nos decía que $d \leq n - k + 1$ obtenemos la igualdad.

□

También existen los códigos Reed-Solomon generalizados, los cuales, se corresponden con un subconjunto de los códigos de Reed-Solomon al depender el código de un elemento $b \in \mathbb{F}_q^n$.

Definición 3.4. Un código Reed-Solomon generalizado es un código sobre un cuerpo finito $\mathbb{F}_q$, con un vector $b = (b_1, ..., b_n) \in \mathbb{F}_q^n$, donde $b_i \neq 0 \, \forall i$ y con un vector $a = (a_1, ..., a_n) \in \mathbb{F}_q^n$, donde $a_i \neq a_j \, \forall i \neq j$, y n es un número entero con $0 < n \leq q - 1$.

Considerando igual que para los códigos Reed-Solomon $L_k \subset \mathbb{F}_q$, el espacio vectorial de dimensión k formado por todos los polinomios de grado menor que k, tenemos que el código Reed-Solomon asociado al vector a y b, denotado por $GRS_k(a, b)$ es:

$$GRS_k(a, b) = \{(b_1 f(a_1), ..., b_n f(a_n)) \mid f \in L_k\}$$

Nota 3.5. Podemos observar que si tomamos el vector $b = (1, ..., 1) \in \mathbb{F}_q^n$, se cumple por construcción que $GRS_k(a, b) = RS_k(a)$. Por tanto las propiedades que vemos a continuación para los códigos de Reed-Solomon generalizados son totalmente validas para los códigos Reed-Solomon.

Proposición 3.6. *El código dual de $GRS_k(a, b)$ es $GRS_{n-k}(a, b')$, donde $b' = (b'_1, ..., b'_n)$ es un vector de $\mathbb{F}_q^n$ con $b'_i \neq 0 \, \forall i = 1, ..., n$. Además, $b' \in GRS_{n-1}(a, b)^\perp$ y se cumple que*

$$\sum_{i=1}^{n} b'_i b_i f(a_i) = 0 \ \forall f \in L_{k-1} \tag{3.1}$$

Demostración. Si tomamos $k = n - 1$ tenemos que $GRS_{n-1}(a, b)^\perp$ es un código $[n, 1, n]$ MDS al ser $GRS_{n-1}(a, b)$ un código MDS y este tiene como base $b' = (b'_1, ..., b'_n)$. Como el código $GRS_1(a, b')$ coincide con el espacio vectorial generado por el vector b', coincide también con el dual, es decir, $< b' >= GRS_1(a, b') = GRS_{n-1}(a, b)^\perp$. Además, si tomamos $f \in L_{k-1}$ tenemos que el vector $bf(a) = (b_1 f(a_1), ..., b_n f(a_n)) \in GRS_{n-1}(a, b)$, y al ser el vector $b' \in GRS_1(a, b')$, entonces su producto sera nulo y por ello se cumple (3.1).

Si tomamos $0 \leq k \leq n - 1$, $f \in L_k$ y $g \in L_{n-k}$ tenemos que $h = fg \in L_{n-1}$ y aplicando (3.1) tenemos que $\sum_{i=1}^{n} b'_i b_i f(a_i) g(a_i) = 0$. Por tanto $GRS_k(a, b)^\perp \subseteq GRS_{n-k}(a, b')$ y como $GRS_k(a, b)^\perp$ tiene dimension $n - k$ entonces se da la igualdad.

□

Corolario 3.7. *El código dual de $GRS_k(a, b)$ es MDS.*

Demostración. Sabemos que el dual de un código MDS es también un código MDS.

□

3.2. Matriz generatriz y de control de los códigos Reed-Solomon y Reed-Solomon generalizados

Una base para L_k nos proporciona una base del código $RS_k(a)$ y $GRS_k(a,b)$ al ser estos códigos lineales. Si tomamos al igual que antes, $\{1, x, ..., x^{k-1}\}$ como base de L_k, tenemos que la matriz generatriz se obtiene evaluando cada polinomio de la base sobre las coordenadas del vector a (y si se trata del caso generalizado, multiplicando además por la coordenada correspondiente de b), lo cual nos proporciona una fila de la matriz generatriz.

Por tanto la matriz generatriz, $G \in \mathbb{F}_q^{k\times n}$, de un código Reed-Solomon generalizado es:

$$G = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ b_1a_1 & b_2a_2 & \cdots & b_na_n \\ \vdots & \vdots & & \vdots \\ b_1a_1^{k-1} & b_2a_2^{k-1} & \cdots & b_na_n^{k-1} \end{pmatrix}$$

Como hemos dicho, si el elemento $b \in \mathbb{F}_q^n$ es $b = (1, ..., 1)$ entonces tenemos que $GRS_k(a,b) = RS_k(a)$. Por tanto la matriz generatriz de un Reed-Solomon será igual que la matriz generatriz del Reed-Solomon generalizado, tomando en dicha matriz el valor 1 en todas las coordenadas del vector b, es decir, $b_i = 1\ \forall i = 1, ..., n$.

En cuanto a la matriz de control de los códigos de Reed-Solomon y Reed-Solomon generalizados, al ser dichos códigos lineales, se obtiene según lo visto en la definición 2.11

3.3. Codificación y descodificación de los códigos Reed-Solomon y Reed-Solomon generalizados

Para realizar la codificación de un mensaje $m = (m_1, ..., m_k)$, de tamaño k, empleando un código Reed-Solomon generalizado $GRS_k(a,b)$, tenemos que obtener su polinomio: $f_m(x) = \sum_{i=1}^{k} m_i x^{i-1}$. Ahora, evaluando dicho polinomio sobre las coordenadas del vector a y multiplicando el resultado obtenido de cada evaluación por la coordenada correspondiente del vector b obtenemos su codificación: $c = (b_1 f_m(a_1), ..., b_n f_m(a_n))$. Obviamente, podemos observar que esto coincide con realizar la codificación explicada en 2.4, para cualquier código lineal, realizando simplemente la multiplicación del mensaje por la matriz generatriz, es decir, $c = mG$.

Para el proceso de descodificación, suponemos que se envía el bloque de mensaje $c \in GRS_k(a,b)$ y recibimos el mensaje $y = c + e$, donde el peso del vector de error es menor que la mitad de la distancia del código $GRS_k(a,b)$, d, es decir, $\omega(e) \leq t = \frac{d-1}{2}$.

Vamos a explicar un método general válido para cualquier código Reed-Solomon y Reed-Solomon generalizado conocido como algoritmo de Berlekamp-Welch y otro método válido solamente para códigos Reed-Solomon y Reed-Solomon generalizados que sean cíclicos, el cual presenta un coste computacional menor ya que, a pesar de tener que calcular $n-k$ síndromes, resolver dos sistemas con $\omega(e)$ incógnitas cada uno y obtener $\omega(e)$ raíces, como veremos a continuación, tenemos que tener en cuenta que siempre el parámetro $\omega(e)$ será notablemente inferior al parámetro n y existen técnicas que permiten obtener rápidamente los síndromes cuando trabajamos con códigos cíclicos.

En primer lugar explicamos el método de Berlekamp-Welch y en segundo lugar el método exclusivo para códigos Reed-Solomon y Reed-Solomon generalizados cíclicos:

1 Para el primer método, denotamos por $I = \{i \in \{1,...,n\} \mid b_i f(a_i) \neq y_i\}$ las posiciones donde existen errores, podemos suponer que $|I| = t$, y $E(x) = \prod_{i \in I}(x - a_i)$ es el polinomio mónico de grado t. Entonces $E(a_i)b_i f(a_i) = E(a_i)y_i \; \forall i = 1,...,n$, ya que $E(a_i) = 0$ si $i \in I$ y si no se puede simplificar la ecuación obteniendo $b_i f(a_i) = y_i \; \forall i \notin I$. Podemos expresar los polinomios como $E(x) = x^t + A_{t-1}x^{t-1} + ... + A_0$ y $E(x)f(x) = B_{t+k-1}x^{t+k-1} + ... + B_0$ donde $A_i, B_j \in \mathbb{F}_q$ son desconocidas $\forall i = 0,...,t-1 \; j = 0,...,t+k-1$. Por tanto, resolviendo el sistema anterior, formado por n ecuaciones y $2t+k$ incógnitas (que tiene solución al ser $t < \frac{d-1}{2}$) obtendríamos el polinomio $E(x)$ y por tanto las posiciones donde se encuentra el error, junto con su valor tras obtener las raíces de $E(x)$.

2 Para el segundo método, denotamos $y = (y_1,...,y_n)$ y entonces tenemos $y(x) = y_n x^{n-1} + ... + y_2 x + y_1$. Ahora vamos a calcular el síndrome $S = (S_1,...,S_{n-k})$ sabiendo que si H es la matriz de control del código entonces, $S = Hy^T$, o lo que es igual al ser un código cíclico, que $S_i = y(\beta^i) = e(\beta^i) \; \forall i = 1,...,n-k$, siendo β un elemento primitivo de $\mathbb{F}_q$.

Una vez calculados los valores del síndrome, tenemos que calcular el polinomio localizador de errores, $L(x)$ y para ello emplearemos el siguiente teorema:

Teorema 3.8. *Sea $L(x)$ un polinomio localizador de errores de un código Reed-Solomon. Entonces $L(x) = L_{t+1}x^t + ... + +L_2 x + L_1$ se puede obtener resolviendo el siguiente sistema:*

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_{t+1} \\ S_2 & S_3 & \cdots & S_{t+2} \\ \vdots & \vdots & \ddots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t} \end{pmatrix} \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_{t+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Una vez que tenemos calculado el polinomio localizador de errores, $L(x)$, tenemos que obtener sus raíces, para lo cual, si el cuerpo finito no es muy extenso se puede probar con todos los elementos del cuerpo, donde para reducir su costo computacional, se puede aplicar el método de Chien, el cual consiste en tomar cada elemento del cuerpo expresado en forma de potencia de un generador de dicho cuerpo. Aun así, si el cuerpo finito es demasiado extenso, dichas raíces se pueden obtener a partir de un algoritmo probabilístico eficiente conocido como *Equal-degree factorization* cuyo funcionamiento se puede ver en [23] .

Entonces, dichas raíces serán una cierta potencia de nuestro elemento primitivo, β. Si $\beta^{i_1}, ..., \beta^{i_t}$ son estas raíces, tenemos que el polinomio del error será $e(x) = e_{i_1}x^{i_1} + ... + e_{i_t}x^{i_t}$, cuyos valores podemos calcular resolviendo el sistema $S = Hy^T = He^T$, siendo H la matriz de control del código.

Una vez que hemos obtenido el bloque de mensaje libre de errores, $c \in GRS_k(a, b)$, para obtener la descodificación de dicho bloque, m, tenemos que resolver el sistema $c = mG$, donde conocemos todo salvo m.

Ejemplo 3.1. Consideramos el cuerpo finito $\mathbb{F}_{16} = \mathbb{F}_{2^4}$ y tomamos $n = q - 1 = 16 - 1 = 15$ para obtener la máxima longitud posible para el código y que este sea cíclico. Para este ejemplo vamos a considerar que el código es capaz de corregir hasta 3 errores, es decir, la capacidad correctora del código, t, es 3. Como hemos visto que los Reed-Solomon son MDS, tenemos que $n - k = 2t = d - 1$, luego la dimension del código, k, para los parámetros propuestos tiene que ser $k = n - 2t = 9$.

Tomamos como polinomio primitivo para generar $\mathbb{F}_{2^4}$ a $p(x) = x^4 + x + 1$ (Se podría razonar de igual forma con los otros polinomios primitivos de grado 4: $p(x) = x^4 + x^3 + 1$ ó $p(x) = x^4 + x^3 + x^2 + x + 1$).

Vamos a buscar un elemento primitivo de $\mathbb{F}_{2^4}$, α, para ello, consideramos primero $x = \alpha$ y veamos que tiene orden 15:

El orden de un elemento tiene que dividir al orden del grupo, entonces $ord(\alpha)$ tendría que ser 1, 3, 5, o 15 y como el orden de α no es ni 1 ni 3 ni 5 ya que $\alpha \neq 1$, $\alpha^3 \neq 1$ y $\alpha^5 = \alpha^4\alpha = (1 + \alpha)\alpha = \alpha + \alpha^2 \neq 1$, necesariamente su orden es 15 y por tanto es un elemento primitivo de $\mathbb{F}_{2^4}$.

Entonces los elementos del cuerpo se pueden representar como $\{0, 1, \alpha, ..., \alpha^{14}\}$. Calculando todas las potencias, podemos obtener los elementos del cuerpo también como sumas de elementos $\lambda\alpha^i$, con $i \in \{0, 1, 2, 3\}$ y $\lambda \in \mathbb{F}_2$:

$\alpha^6 = \alpha^3 + \alpha^2$, $\alpha^7 = \alpha^4 + \alpha^3 = (1 + \alpha) + \alpha^3 = 1 + \alpha + \alpha^3$, $\alpha^8 = \alpha^2 + 1$, $\alpha^9 = \alpha^3 + \alpha$, $\alpha^{10} = \alpha^2 + x + 1$, $\alpha^{11} = \alpha^3 + \alpha^2 + \alpha$, $\alpha^{12} = \alpha^3 + \alpha^2 + \alpha + 1$, $\alpha^{13} = \alpha^3 + \alpha^2 + 1$, $\alpha^{14} = \alpha^3 + 1$, $\alpha^{15} = 1$.

Entonces tomando como vector $a = (1, \alpha, \alpha^2, ..., \alpha^{14})$, tenemos que el código Reed-Solomon que vamos a emplear es $RS_9(a)$.

Si tomamos $m = (1, 0, 0, 0, 1, 0, 0, 0, 1)$ como el bloque de mensaje que queremos codificar, calculamos su polinomio, $f_m(x) = x^8 + x^4 + 1$, entonces su codificación es:

$$C(m) = (f_m(1), f_m(\alpha), ..., f_m(\alpha^{14})$$

.

Vamos a obtener estos valores:

$f_m(1) = 1 + 1 + 1 = 1$
$f_m(\alpha) = \alpha^8 + \alpha^4 + 1 = \alpha^2 + 1 + \alpha + 1 + 1 = \alpha^2 + \alpha + 1 = \alpha^{10}$
$f_m(\alpha^2) = \alpha^{16} + \alpha^8 + 1 = \alpha + \alpha^2 + 1 + 1 = \alpha^2 + \alpha = \alpha^5$
$f_m(\alpha^3) = \alpha^{24} + \alpha^{12} + 1 = \alpha + \alpha^3 + 1 + \alpha + \alpha^2 + \alpha^3 + 1 = \alpha^2$
$f_m(\alpha^4) = \alpha^{32} + \alpha^{16} + 1 = \alpha^2 + \alpha + 1 = \alpha^{10}$
$f_m(\alpha^5) = \alpha^{40} + \alpha^{20} + 1 = 1 + \alpha + \alpha^2 + \alpha + \alpha^2 + 1 = 0$
$f_m(\alpha^6) = \alpha^{48} + \alpha^{24} + 1 = \alpha^3 + \alpha + \alpha^3 + 1 = \alpha + 1 = \alpha^4$
$f_m(\alpha^7) = \alpha^{56} + \alpha^{28} + 1 = \alpha + \alpha^2 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + 1 = \alpha$
$f_m(\alpha^8) = \alpha^{64} + \alpha^{32} + 1 = 1 + \alpha + \alpha^2 + 1 = \alpha + \alpha^2 = \alpha^5$
$f_m(\alpha^9) = \alpha^{72} + \alpha^{36} + 1 = 1 + \alpha + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^3 + 1 = \alpha$
$f_m(\alpha^{10}) = \alpha^{80} + \alpha^{40} + 1 = \alpha + \alpha^2 + 1 + \alpha + \alpha^2 + 1 = 0$
$f_m(\alpha^{11}) = \alpha^{88} + \alpha^{44} + 1 = 1 + \alpha^2 + \alpha^3 + 1 + \alpha^3 + 1 = \alpha^2 + 1 = \alpha^8$
$f_m(\alpha^{12}) = \alpha^{96} + \alpha^{48} + 1 = \alpha^2 + \alpha^3 + \alpha^3 + 1 = \alpha^2 + 1 = \alpha^8$
$f_m(\alpha^{13}) = \alpha^{104} + \alpha^{52} + 1 = 1 + \alpha^3 + 1 + \alpha + \alpha^3 + 1 = 1 + \alpha = \alpha^4$
$f_m(\alpha^{14}) = \alpha^{112} + \alpha^{56} + 1 = \alpha^{11} + \alpha^7 + 1 = \alpha^3 + \alpha^2 + \alpha + \alpha^3 + \alpha + 1 + 1 = \alpha^2$

A partir de estos valores obtenemos el bloque de mensaje codificado, el cual es $C(m) = (1, \alpha^{10}, \alpha^5, \alpha^2, \alpha^{10}, 0, \alpha^4, \alpha, \alpha^5, \alpha, 0, \alpha^8, \alpha^8, \alpha^4, \alpha^2)$.

Una vez que hemos obtenido el bloque de mensaje codificado, suponemos que enviamos dicho bloque y durante su transmisión se producen tres errores, $\omega(e) = 3$, cuyas posiciones son $0, 7$ y 14:

$$y = c + e = (\boldsymbol{\alpha}, \alpha^{10}, \alpha^5, \alpha^2, \alpha^{10}, 0, \alpha^4, \boldsymbol{\alpha^3}, \alpha^5, \alpha, 0, \alpha^8, \alpha^8, \alpha^4, \boldsymbol{\alpha^{10}})$$

Vamos a calcular el síndrome para poder corregir los errores:

Escribimos y en forma polinomica y obtenemos $y(x) = \alpha + \alpha^{10}x + \alpha^5x^2 + \alpha^2x^3 + \alpha^{10}x^4 + \alpha^4x^6 + \alpha^3x^7 + \alpha^5x^8 + \alpha x^9 + \alpha^8x^{11} + \alpha^8x^{12} + \alpha^4x^{13} + \alpha^{10}x^{14}$. A partir de $y(x)$ vamos a obtener todos los valores del síndrome $S = (S_1, ..., S_6)$:

$S_1 = y(\alpha) = \alpha + \alpha^{11} + \alpha^7 + \alpha^5 + \alpha^{14} + \alpha^{10} + \alpha^{10} + \alpha^{13} + \alpha^{10} + \alpha^4 + \alpha^5 + \alpha^2 + \alpha^9 = 1 + x^3 = \alpha^{14}$
$S_2 = y(\alpha^2) = \alpha + \alpha^{12} + \alpha^9 + \alpha^8 + \alpha^{18} + \alpha^{16} + \alpha^{17} + \alpha^{21} + \alpha^{19} + \alpha^{30} + \alpha^{32} + \alpha^{30} + \alpha^{38} = x = \alpha$
$S_3 = y(\alpha^3) = \alpha + \alpha^{13} + \alpha^{11} + \alpha^{11} + \alpha^{22} + \alpha^{22} + \alpha^{24} + \alpha^{29} + \alpha^{28} + \alpha^{41} + \alpha^{44} + \alpha^{43} + \alpha^{52} = 0$
$S_4 = y(\alpha^4) = \alpha + \alpha^{14} + \alpha^{13} + \alpha^{14} + \alpha^{26} + \alpha^{28} + \alpha^{31} + \alpha^{37} + \alpha^{37} + \alpha^{52} + \alpha^{56} + \alpha^{56} + +\alpha^{66} = 1 + x^3 = \alpha^{14}$
$S_5 = y(\alpha^5) = \alpha + \alpha^{15} + \alpha^{15} + \alpha^{17} + \alpha^{30} + \alpha^{34} + \alpha^{38} + \alpha^{45} + \alpha^{46} + \alpha^{63} + \alpha^{68} + \alpha^{69} + \alpha^{80} = 1 + x = \alpha^4$
$S_6 = y(\alpha^6) = \alpha + \alpha^{16} + \alpha^{17} + \alpha^{20} + \alpha^{34} + \alpha^{40} + \alpha^{45} + \alpha^{53} + \alpha^{55} + \alpha^{74} + \alpha^{80} + \alpha^{82} + \alpha^{94} = x = \alpha$

Una vez obtenidos todos los valores del síndrome, vamos a utilizar estos valores para obtener el polinomio localizador de errores, $L(x) = L_4x^3 + L_3x^2 + L_2x + L_1$, resolviendo el siguiente sistema:

$$\begin{pmatrix} \alpha^{14} & \alpha & 0 & \alpha^{14} \\ \alpha & 0 & \alpha^{14} & \alpha^4 \\ 0 & \alpha^{14} & \alpha^4 & \alpha \end{pmatrix} \begin{pmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Obtenemos, sin tener en cuenta la solución idénticamente nula, $L_i = 0 \, \forall i = 1, 2, 3, 4$, que $L_1 = \alpha^6$, $L_2 = \alpha^{11}$, $L_3 = \alpha^4$ y $L_4 = 1$.

Luego $L(x) = x^3 + \alpha^4x^2 + \alpha^{11}x + \alpha^6$ y sus raíces son $x = 1$, $x = \alpha^7$ y $x = \alpha^{14}$

Por tanto, el polinomio de error es $e(x) = e_3x^{14} + e_2x^7 + e_1$. Ahora resolvemos el siguiente sistema para obtener los valores e_1, e_2, e_3:

$$\begin{pmatrix} 1 & \alpha^7 & \alpha^{14} \\ 1 & \alpha^{14} & \alpha^{13} \\ 1 & \alpha^6 & \alpha^{12} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} \alpha^{14} \\ \alpha \\ 0 \end{pmatrix}$$

Obtenemos que $e_1 = \alpha^4, e_2 = \alpha^9, e_3 = \alpha^4$, luego $e(x) = \alpha^4x^{14} + \alpha^9x^7 + \alpha^4$ y entonces podemos recuperar el mensaje codificado sin errores: $c = y + e = (\alpha + \alpha^4, \alpha^{10}, \alpha^5, \alpha^2, \alpha^{10}, 0, \alpha^4, \alpha^3 + \alpha^9, \alpha^5, \alpha, 0, \alpha^8, \alpha^8, \alpha^4, \alpha^{10} + \alpha^4) = (1, \alpha^{10}, \alpha^5, \alpha^2, \alpha^{10}, 0, \alpha^4, \alpha, \alpha^5, \alpha, 0, \alpha^8, \alpha^8, \alpha^4, \alpha^2)$.

Por ultimo, resolvemos el siguiente sistema que nos permite recuperar el polinomio, $f(x) = a_{15}x^{14} + ... + a_2x + a_1$, que se corresponde con el mensaje descodificado:

$$(a_1, a_2, \cdots, a_9) \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^{14} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^9 & \cdots & (\alpha^9)^{14} \end{pmatrix} = \left(1, \alpha^{10}, \cdots, \alpha^2\right)$$

Para resolver dicho sistema basta con emplear en este caso 9 columnas linealmente independientes de la matriz generatriz.

Los valores que se obtiene al resolver este sistema son $a_1 = 1$, $a_5 = 1$, $a_9 = 1$ y $a_i = 0 \, \forall i = 2, 3, 4, 6, 7, 8$ y por tanto el mensaje sería $m = (1, 0, 0, 0, 1, 0, 0, 0, 1)$, que se corresponde con el mensaje del que partíamos.

La principal desventaja se encuentra en que la longitud de los códigos de Reed-Solomon y Reed-Solomon generalizados (sobre el cuerpo $\mathbb{F}_q$) es de un máximo de $q - 1$ y por ello no existen códigos de Reed-Solomon y Reed-Solomon generalizados sobre el cuerpo $\mathbb{F}_2$. Como dicha longitud se encuentra condicionada al tamaño del cuerpo finito, es necesario emplear cuerpos finitos grandes.

Debido a esto, surgen códigos alternativos que emplean cuerpos finitos pequeños y mantienen las mismas buenas características que tenían los códigos de Reed-Solomon y Reed-Solomon generalizados, como por ejemplo los códigos de Goppa.

Capítulo 4

Códigos de Goppa

4.1. Introducción

En 1970, V.D. Goppa creó unos códigos que actualmente tienen bastante interés en la criptografía, los cuales eran conocidos por muchos autores como códigos casi aleatorios aunque son normalmente llamados códigos de Goppa en su honor.

Algunas de sus propiedades es que hay una gran cantidad de códigos de Goppa con parámetros fundamentales muy parecidos, que son difíciles de distinguir de otros códigos lineales aleatorios, se conocen algoritmos tanto de codificación como de descodificación para dichos códigos y son relativamente sencillos de construir.

Es por ello que McEliece eligió los códigos de Goppa para introducir su criptosistema en 1978, el cual mostraremos en el capítulo 5. En este tema definiremos dichos códigos y un algoritmo de descodificación para ellos entre otras propiedades.

Definición 4.1. Sea $L = (\alpha_0, ..., \alpha_{n-1}) \in \mathbb{F}_{q^m}^n$ con $\alpha_i \neq \alpha_j \ \forall i \neq j$ y $g(x)$ un polinomio de grado t con coeficientes en $\mathbb{F}_{q^m}$, de forma que $g(\alpha_i) \neq 0 \ \forall \ \alpha_i \in L$. Llamaremos *código de Goppa* correspondiente a L y $g(x)$, denotado por $\Gamma(L, g)$, al código corrector de errores formado por todos los vectores $c = (c_0, ..., c_{n-1}) \in \mathbb{F}_q^n$ que cumplen que:

$$R_c(x) = \sum_{i=1}^{n} \frac{c_{i-1}}{x - \alpha_{i-1}} \equiv 0 \mod g(x)$$

Nota 4.2. Si el polinomio de Goppa, $g(x)$, es irreducible entonces el código de Goppa $\Gamma(L, g)$ se dice que es irreducible y se verifica de forma trivial que $g(\alpha_i) \neq 0$ ya que si α_i fuera una raíz de $g(x)$, entonces $g(x)$ sería reducible .

Nota 4.3. Si $g(x)$ y $f(x)$ son polinomio de Goppa de cierto grado con coeficientes en $\mathbb{F}_q$ de los códigos $\Gamma(L, g)$ y $\Gamma(L, f)$ respectivamente. Se verifica que si $f(x)|g(x)$ entonces

$\Gamma(L,g) \subseteq \Gamma(L,f)$. Esto es evidente ya que si $c \in \Gamma(L,g)$ se tiene que $R_c(x) \equiv 0 \bmod g(x)$ y como $f(x)|g(x)$ entonces $R_c(x) \equiv 0 \bmod f(x)$.

Nota 4.4. Sea $p_i(x) = p_{i,0} + p_{i,1}x + ... + p_{i,t-1}x^{t-1}$ congruente con $\dfrac{1}{x-\alpha_i}$ modulo $g(x)$ $\forall i = 0, ..., n-1$, entonces un vector $c = (c_0, ..., c_{n-1}) \in \Gamma(L,g)$ si y solo si tenemos que $\sum_{i=1}^{n} c_{i-1}p_{i-1,j} = 0 \ \forall j = 1, ..., t-1$.

Además, tenemos que $p_i(x) \equiv -g(\alpha_i)^{-1}\dfrac{g(x)-g(\alpha_i)}{x-\alpha_i} \bmod g(x)$ porque:

1) Si denotamos $d_i(x) = g(x) - g(\alpha_i)$ y $g(x) = g_0 + g_1x + ... + g_tx^t$, podemos observar que α_i es raíz de $d_i(x)$ y como también lo es de $x - \alpha_i$ entonces $x - \alpha_i \mid d_i(x)$ $\forall i = 0, ..., n-1$

2) Multiplicando por $x - \alpha_i$ tenemos que:

$$p_i(x)(x-\alpha_i) = -(g(x) - g(\alpha_i))g(\alpha_i)^{-1} = 1 - g(x)g(\alpha_i)^{-1} \equiv 1 \bmod g(x)$$

Proposición 4.5. *Según las consideraciones vistas en la nota 4.4, una matriz de control para un código de Goppa $\Gamma(L,g)$ es:*

$$H' = \left(g(\alpha_0)^{-1}\frac{d_0}{x-\alpha_0}, ..., g(\alpha_{n-1})^{-1}\frac{d_{n-1}}{x-\alpha_{n-1}} \right)$$

Nota 4.6. Como en la matriz H' tenemos que cada término $g(\alpha_i)^{-1}\dfrac{d_i}{x-\alpha_i}$ hace referencia a una columna de la matriz, es decir, calculando $\dfrac{d_i}{x-\alpha_i} = g_t(x^{t-1} + x^{t-2}\alpha_i + ... + \alpha_i^{t-1}) + ... + g_2(x+\alpha_i) + g_1$ tenemos que dicha matriz es:

$$H' = \begin{pmatrix} g(\alpha_0)^{-1}g_t & \cdots & g(\alpha_{n-1})^{-1}g_t \\ g(\alpha_0)^{-1}(g_{t-1} + g_t\alpha_0) & \cdots & g(\alpha_{n-1})^{-1}(g_{t-1} + g_t\alpha_{n-1}) \\ \vdots & & \vdots \\ g(\alpha_0)^{-1}(g_1 + g_2\alpha_0 + ... + g_t\alpha_0^{t-1}) & \cdots & g(\alpha_{n-1})^{-1}(g_1 + g_2\alpha_{n-1} + ... + g_t\alpha_{n-1}^{t-1}) \end{pmatrix}$$

Ahora si denotamos las matrices E y H como:

$$E = \begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix}, \quad H = \begin{pmatrix} g(\alpha_0)^{-1} & \cdots & g(\alpha_{n-1})^{-1} \\ g(\alpha_0)^{-1}\alpha_0 & \cdots & g(\alpha_{n-1})^{-1}\alpha_{n-1} \\ \vdots & & \vdots \\ g(\alpha_0)^{-1}\alpha_0^{t-1} & \cdots & g(\alpha_{n-1})^{-1}\alpha_{n-1}^{t-1} \end{pmatrix}$$

Podemos ver que nuestra matriz H' se puede expresar como $H' = E \cdot H$. Como la matriz E es invertible, al ser $det(E) = g_t^t \neq 0$, ya que si g_t fuera 0 nuestro polinomio $g(x)$ tendría grado $t-1$ y por definición sabemos que tiene grado t. Entonces H es también una matriz de control para $\Gamma(L, g)$.

Proposición 4.7. *Sea el código de Goppa $\Gamma(L, g)$. Tenemos que:*

a) $\Gamma(L, g)$ es un código lineal.

b) La longitud de las palabras de código $c \in \Gamma(L, g)$ es $n = |L|$.

c) La dimension del código $\Gamma(L, g)$ sobre $\mathbb{F}_q$ es $k \geq n - mt$

d) La distancia mínima de $\Gamma(L, g)$ es $d \geq t + 1$

Demostración.

a) Sean $a = (a_0, ..., a_{n-1})$, $b = (b_0, ..., b_{n-1}) \in \Gamma(L, g)$. Entonces tenemos que $R_a(x) = \sum_{i=1}^{n} \frac{a_{i-1}}{x - \alpha_i} \equiv 0 \mod g(x)$ y $R_b(x) = \sum_{i=1}^{n} \frac{b_{i-1}}{x - \alpha_i} \equiv 0 \mod g(x)$ y como para el vector $c = a + b$ tenemos que

$$R_c(x) = \sum_{i=1}^{n} \frac{a_{i-1} + b_{i-1}}{x - \alpha_i} = \sum_{i=1}^{n} \frac{a_{i-1}}{x - \alpha_i} + \sum_{i=1}^{n} \frac{b_{i-1}}{x - \alpha_i} \equiv 0 \bmod g(x)$$

al ser tanto $R_a(x)$ como $R_b(x)$ congruentes a 0 modulo $g(x)$ su suma también y entonces $c \in \Gamma(L, g)$.

De igual forma, si tomamos $d \in \mathbb{F}_q$ tenemos que, se cumple también de forma trivial, que para el vector $c = (da_0, ..., da_{n-1})$ tenemos que $R_c(x) \equiv 0$ y por tanto también $c \in \Gamma(L, g)$.

b) Evidente.

c) Como tanto los α_i como los $g(\alpha_i)^{-1}$ son elementos de $\mathbb{F}_{q^m}$, si sustituimos cada uno de estos por el vector correspondiente de tamaño m formado por elementos de $\mathbb{F}_q$ tendremos la matriz H formada por elementos de $\mathbb{F}_q$ con mt filas. Por ello, $k = n - mt$ si todas las filas son linealmente independientes. Si no son linealmente independientes, tenemos que $k > n - mt$.

d) Como sabemos que t columnas de la matriz H son linealmente independientes tenemos que la distancia será al menos $t + 1$ por lo visto en la proposición 2.17 .

□

Por tanto, la distancia mínima de los códigos de Goppa es mayor que $t+1$, aunque si nos restringimos sobre ciertos polinomios y el cuerpo $\mathbb{F}_2$ podemos garantizar una distancia mínima de al menos $2t+1$, lo cual resulta de gran utilidad ya que por ejemplo, permite doblar la capacidad correctora de estos códigos.

Proposición 4.8. *Si el código de Goppa $\Gamma(L,g)$ se define sobre $\mathbb{F}_2$, con $L=(\alpha_0,...,\alpha_{n-1})\in \mathbb{F}_{2^m}^n$ y el polinomio de Goppa $g(x)\in\mathbb{F}_{2^m}[x]$ de grado t es separable, es decir, el polinomio tiene todas sus raíces simples, entonces la distancia mínima del código $d\geq 2t+1$.*

Demostración. Para ello basta ver que $\Gamma(L,g)=\Gamma(L,g^2)$ ya que tenemos que la distancia mínima de $\Gamma(L,g^2)$ es mayor o igual que $2t+1$ por ser $g^2(x)$ un polinomio de grado $2t$. Veamos la doble contención:

$\supset$) Sea $c\in\Gamma(L,g^2)$, entonces tenemos que $R_c(x)\equiv 0 \bmod g^2(x)$. Como $g(x)|g^2(x)$ es evidente que $R_c(x)\equiv 0 \bmod g(x)$, entonces $\Gamma(L,g^2)\subseteq\Gamma(L,g)$.

$\subset$) Sea $c\in\Gamma(L,g)$. Definimos $f(x)=\prod_{i=0}^{n-1}(x-\alpha_i)^{c_i}$ y como su derivada es $f'(x)=\sum_{i=0}^{n-1}(c_i(x-\alpha_i)^{c_i-1}\prod_{j=0,j\neq i}^{n-1}(x-\alpha_j)^{c_j})$ tenemos que $\frac{f'(x)}{f(x)}=\sum_{i=0}^{n-1}\frac{c_i}{x-\alpha_i}\equiv 0 \bmod g(x)$ por ser c una palabra del código de Goppa. Como $f(x)$ y $g(x)$ no tienen factores comunes ya que las raíces de $f(x)$ son por definición $\alpha_i\ \forall i=0,...,n-1$ y $g(\alpha_i)\neq 0$ $\forall i=0,...,n-1$, entonces $g(x)|f'(x)$. Ademas como en el cuerpo $\mathbb{F}_{2^m}$ las derivadas solo tienen potencias pares debido a que al derivar una potencia par el coeficiente será congruente con 0 en $\mathbb{F}_2$. Por ello, podemos expresar $f'(x)$ como $f'(x)=h^2(x)$ para algún polinomio $h(x)$. Luego $g(x)|h^2(x)$, pero como $g(x)$ es separable entonces $g(x)|h(x)$ y también $g^2(x)|h^2(x)$. Debido a esto $\frac{f'(x)}{f(x)}\equiv 0 \bmod g^2(x)$ y entonces $\Gamma(L,g)\subseteq\Gamma(L,g^2)$. □

4.2. Descodificación

Si consideramos una palabra del código $c=(c_0,...,c_{n-1})\in\Gamma(L,g)$, que enviamos a un receptor, el cual recibe $y=(y_0,...,y_{n-1})=c+e$ donde $e=(e_0,...,e_{n-1})$ es el vector de error que surge respecto la palabra c, donde suponemos que $w(e)\leq r$, es decir, que el peso del error es menor que la capacidad correctora que tiene el código $\Gamma(L,g)$ ($r=t$ si cumple las hipótesis de la proposición 4.8 o $r=t/2$ en otro caso).

Para la corrección de los errores producidos, vamos a introducir los conceptos de síndrome y polinomio localizador y evaluador de errores para los códigos de Goppa.

Definición 4.9. Con las consideraciones anteriores, llamamos síndrome del vector $y = (y_0, ..., y_{n-1})$ y lo denotamos $s(x)$ al valor:

$$s(x) = \sum_{i=0}^{n-1} \frac{y_i}{x - \alpha_i} (\mathrm{mod}\, g(x)) = \sum_{i=0}^{n-1} \frac{e_i}{x - \alpha_i} (\mathrm{mod}\, g(x))$$

Definición 4.10. Llamamos polinomios localizador y evaluador de errores, denotados respectivamente por $L(x)$ y $E(x)$ a los polinomios dados de la forma:

$$L(x) = \prod_{i|e_i \neq 0} (x - \alpha_i) \ , \ E(x) = \sum_{i|e_i \neq 0} e_i \prod_{j|e_j \neq 0, j \neq i} (x - \alpha_j)$$

Nota 4.11. Si los códigos de Goppa empleados son binarios, bastaría con calcular el polinomio localizador de errores, $L(x)$, ya que una vez calculadas las posiciones en que se encuentra el error, como los únicos valores posibles son el 0 y 1, simplemente habría que cambiar en dichas posiciones el valor que tenemos.

Pero si estamos empleando códigos de Goppa no binarios, necesitamos calcular tanto el polinomio localizador como corrector de errores, lo cual no complica demasiado la codificación al cumplirse que $s(x)L(x) \equiv E(x) \bmod g(x)$, siendo $L(x)$ el polinomio que posee menor grado de entre el conjunto de polinomios que cumplen la ecuación anterior.

Esta congruencia se ve fácilmente multiplicando el polinomio del síndrome por el localizador de errores:

$$E(x) = \sum_{i|e_i \neq 0} e_i \prod_{j|e_j \neq 0, j \neq i} (x - \alpha_j) = \sum_{i|e_i \neq 0} \frac{e_i L(x)}{x - \alpha_i} =$$

$$= L(x) \sum_{i=0}^{n-1} \frac{e_i}{x - \alpha_i} (\mathrm{mod}\, g(x)) = s(x)L(x) (\mathrm{mod}\, g(x))$$

Para la obtención de los polinomios localizador y evaluador de errores que nos permitan corregir los errores producidos, aunque existen diversos algoritmos que permiten corregir los errores de palabras de códigos de Goppa construidos sobre cualquier cuerpo, como por ejemplo el algoritmo de Berlekamp-Massey [24] [26] o el algoritmo de Euclides extendido [25] [23], vamos a explicar el algoritmo de Patterson, cuyo uso es exclusivo para códigos de Goppa binarios, ya que solo trabajaremos con códigos de Goppa construidos sobre el cuerpo $\mathbb{F}_{2^m}$, con $m \in \mathbb{N}$.

4.2.1. Algoritmo de Patterson

Dado un código de Goppa binario, $\Gamma(L, g)$, tenemos que $g(x)$ es un polinomio con coeficientes en $\mathbb{F}_{2^m}$ y $L = (\alpha_0, ..., \alpha_{n-1}) \in \mathbb{F}_{2^m}^n$.

El algoritmo consiste primero en calcular el síndrome del vector y visto en la definición 4.9 y a partir del síndrome, calculamos el polinomio localizador de errores $L(x)$, para ello:

- Tenemos que buscar el polinomio $f(x)$ que cumpla que $s(x)f(x) \equiv 1 \bmod g(x)$. Si dicho polinomio es la identidad, es decir, $f(x) = x$ tendríamos que $L(x) = x$ y hemos terminado. Si esto no ocurre, seguimos realizando los pasos siguientes.
- Calculamos el polinomio $h(x)$ tal que $h^2(x) \equiv f(x) + x \bmod g(x)$
- Buscamos los polinomios $a(x)$ y $b(x)$ de grado mínimo que cumplan que $h(x)b(x) \equiv a(x) \bmod g(x)$
- El polinomio localizador de errores será $L(x) = a^2(x) + xb^2(x)$

Ahora, con el polinomio localizador de errores que hemos calculado procedemos a obtener las posiciones en que hay errores $(i \mid e_i \neq 0)$. Para ello, calculamos las raíces del polinomio $L(x)$, las cuales, deben ser elementos del vector L. Entonces, las posiciones de errores se corresponden con las posiciones que ocupan en el vector L los elementos que son raices del polinomio localizador de errores, $L(x)$.

Por tanto, una vez que conocemos el vector de errores $e = (e_0, ..., e_{n-1})$, ya podemos obtener la palabra codificada sin errores: $c = y - e = y + e$

Ejemplo 4.1. Emplearemos el cuerpo $\mathbb{F}_{2^4}$ construido a partir del polinomio primitivo de grado 4, $p(x) = x^4 + x + 1$, de igual forma que se realizo en el ejemplo 3.1.

Vamos a construir un código de Goppa de longitud $n = 16$. Por tanto, tomaremos como vector $L = (\alpha, \alpha^2, \cdots, \alpha^{14}, 1, 0) \in \mathbb{F}_{2^4}^{16}$, siendo α un elemento primitivo de $\mathbb{F}_{2^4}$, y emplearemos un polinomio de Goppa irreducible, $g(x) = x^3 + \alpha^2 x + \alpha$, para tener directamente que $g(l) \neq 0 \, \forall l \in L$ según la nota 4.2.

Ahora obtenemos la matriz de control del código, H, calculando los elementos inversos de $g(l) \, \forall l \in L$. En este ejemplo, podemos expresar todos los elementos menos el 0 como una cierta potencia del generador α. Por tanto, como $g(l) \neq 0 \, \forall l \in L$, para calcular su elemento inverso tenemos simplemente que expresar el elemento resultante de evaluar sobre el polinomio g como potencia y dividir $\alpha^{15} = 1$ por dicha potencia:

$g(\alpha) = 2\alpha^3 + \alpha = \alpha$
$g(\alpha^2) = \alpha^6 + \alpha^4 + \alpha = \alpha^3 + \alpha^2 + 1 = \alpha^{13}$
$g(\alpha^3) = \alpha^9 + \alpha^5 + \alpha = \alpha^3 + \alpha^2 + a = \alpha^{11}$
$g(\alpha^4) = \alpha^{12} + \alpha^6 + \alpha = 1$
$g(\alpha^5) = \alpha^{15} + \alpha^7 + \alpha = \alpha^3$
$g(\alpha^6) = \alpha^3 + \alpha^8 + \alpha = \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^{12}$
$g(\alpha^7) = \alpha^6 + \alpha^9 + \alpha = \alpha^2$
$g(\alpha^8) = \alpha^9 + \alpha^{10} + \alpha = \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^{12}$
$g(\alpha^9) = \alpha^{12} + \alpha^{11} + \alpha = \alpha + 1 = \alpha^4$
$g(\alpha^{10}) = 1 + \alpha^{12} + \alpha = \alpha^3 + \alpha^2 = \alpha^6$
$g(\alpha^{11}) = \alpha^3 + \alpha^{13} + \alpha = \alpha^2 + \alpha + 1 = \alpha^{10}$
$g(\alpha^{12}) = \alpha^6 + \alpha^{14} + \alpha = \alpha^2 + \alpha + 1 = \alpha^{10}$
$g(\alpha^{13}) = \alpha^9 + 1 + \alpha = \alpha^3 + 1 = \alpha^{14}$
$g(\alpha^{14}) = \alpha^{12}$
$g(1) = 1 + \alpha^2 + \alpha = \alpha^{10}$
$g(0) = \alpha$

Ahora obtenemos fácilmente sus elementos inversos:

$\frac{1}{g(\alpha)} = \alpha^{14}$, $\frac{1}{g(\alpha^2)} = \alpha^2$, $\frac{1}{g(\alpha^3)} = \alpha^4$, $\frac{1}{g(\alpha^4)} = 1$, $\frac{1}{g(\alpha^5)} = \alpha^{12}$, $\frac{1}{g(\alpha^6)} = \alpha^3$, $\frac{1}{g(\alpha^7)} = \alpha^{13}$, $\frac{1}{g(\alpha^8)} = \alpha^3$, $\frac{1}{g(\alpha^9)} = \alpha^{11}$, $\frac{1}{g(\alpha^{10})} = \alpha^9$, $\frac{1}{g(\alpha^{11})} = \alpha^5$, $\frac{1}{g(\alpha^{12})} = \alpha^5$, $\frac{1}{g(\alpha^{13})} = \alpha$, $\frac{1}{g(\alpha^{14})} = \alpha^3$, $\frac{1}{g(1)} = \alpha^5$, $\frac{1}{g(0)} = \alpha^{14}$

Con ellos obtenemos ahora la matriz de control, H, de tamaño 3×16 :

$$H = \begin{pmatrix} \alpha^3+1 & \alpha^2 & \alpha+1 & \cdots & \alpha^3 & \alpha^2+\alpha & \alpha^3+1 \\ 1 & \alpha+1 & \alpha^3+\alpha+1 & \cdots & \alpha^2 & \alpha^2+\alpha & 0 \\ \alpha & \alpha^3+\alpha^2 & \alpha^2+\alpha+1 & \cdots & \alpha & \alpha^2+\alpha & 0 \end{pmatrix}$$

Como estamos trabajando sobre el cuerpo finito $\mathbb{F}_{2^4}$, extension de $\mathbb{F}_2$, si fijamos una base de la extension sobre $\mathbb{F}_2$, cada elemento de $\mathbb{F}_{2^4}$ se puede expresar como un vector de $\mathbb{F}_2^4$ en dicha base. Esta técnica se conoce como descenso de cuerpo.

Consideramos la base $\{1, \alpha, \alpha^2, \alpha^3\}$, entonces si tomamos por ejemplo el elemento $\alpha^3 + 1$ podemos expresarlo como el vector $(1, 0, 0, 1)$

De esta forma, podemos expresar la matriz de control H formada por elementos de $\mathbb{F}_2$:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Ahora obtenemos la matriz generatriz, G, según lo visto en la definición 2.11 y obtenemos:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Por tanto, podemos observar que la dimension de nuestro código de Goppa $\Gamma(L, g)$ es $k = 4$ y la distancia mínima es $d = 7$ al cumplirse las hipótesis de la proposición 4.8 y al tener la primera fila de la matriz generatriz 7 como valor del peso de Hamming, la cual sabemos que se corresponde con una palabra del código $\Gamma(L, g)$. Debido a esto, la capacidad correctora del código es $t = 3$.

Supongamos ahora que queremos enviar a una persona la letra $'H'$ como mensaje. Transformamos dicha letra ASCII en su correspondiente valor como vector de longitud 8 sobre el cuerpo binario y obtenemos que el mensaje que queremos enviar se corresponde con $m = (0, 1, 0, 0, 1, 0, 0, 0)$.

Como nuestro código tiene dimension $k = 4$, es necesario dividir el mensaje en dos bloques de longitud 4 y realizar la codificación de cada bloque por separado según lo visto en la definición 2.4.

Juntando la codificación de los dos bloques obtenemos que el mensaje codificado, de longitud 32, es:

$$c = (0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1)$$

Ahora suponemos, para no extender el ejemplo, que únicamente se han producido 3 errores en el primer bloque, de forma que necesitaremos aplicar el algoritmo de

Patterson solo para el primer bloque ya que, para el segundo bloque obtendremos que el valor de su síndrome es 0 al ser una palabra del código.

Consideramos el vector de errores de tamaño 16, donde los tres errores se producen en las coordenadas 4, 8 y 13, es decir,

$$e = (0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0)$$

y entonces el vector resultante es

$$y = (0,1,0,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,1,0,0,0,0,1,1)$$

donde el primer y segundo bloque son

$$y_1 = (0,1,0,0,0,0,0,0,1,1,0,1,0,1,0,1,1)$$

y

$$y_2 = (1,0,0,0,1,0,1,0,1,1,0,0,0,0,1,1)$$

respectivamente.

Ahora comenzamos a aplicar el algoritmo de Patterson sobre el bloque y_1 para obtener las posiciones de error. Todos los valores necesarios para aplicar este algoritmo se han obtenido a partir de *Sagemath* donde, para el calculo de algunos de ellos, como por ejemplo, el polinomio $h(x)$, se ha empleado un procedimiento que simplifica su cálculo. Tanto el programa realizado en *Sagemath* como todos los procedimientos empleados se muestran en [1, 4.3.2] y [1, 4.3.3] del trabajo fin de grado de informática.

Calculamos primero el síndrome de y_1 visto en la definición 4.9, cuyo valor es:

$$s(y_1) = \sum_{i=0}^{15} \frac{y_{1_i}}{x - l_i} \equiv (\alpha^3 + 1)x^2 + (\alpha^2 + \alpha)x + \alpha \bmod g(x)$$

donde l_i hace referencia a la coordenada i-esima del vector L.

Ahora calculamos el polinomio $f(x)$, el cual es el inverso del síndrome módulo g:

$$f(x) = (\alpha^2 + \alpha)x^2 + \alpha^3 x + \alpha^3 + \alpha$$

Como $f(x) \neq x$ continuamos con los pasos restantes del algoritmo. Por ello, ahora calculamos $h(x)$ de forma que $h^2(x) \equiv f(x) + x \bmod g(x)$:

$$h(x) = (\alpha^3 + 1)x^2 + (\alpha^2 + \alpha)x + \alpha^3 + \alpha^2 + \alpha + 1$$

Ahora obtenemos los polinomios $a(x)$ y $b(x)$ de menor grado:

$$a(x) = (\alpha^2 + \alpha)x + 1,\ b(x) = \alpha x + \alpha^3 + \alpha + 1$$

Por tanto el polinomio localizador de errores resultante es:

$$L(x) = \alpha^2 x^3 + (\alpha^2 + \alpha + 1)x^2 + (\alpha^3 + 1)x + 1$$

Obteniendo entonces las raíces del polinomio L, las cuales se corresponden a los valores α^5, α^9 y α^{14}, obtenemos las posiciones en que ha ocurrido un error.

Entonces, resolviendo ahora un sistema para cada bloque c_1, c_2, teniendo en cuenta la definición 2.4, ya que conocemos tanto G como c_1 y c_2 podemos recuperar el mensaje $m = (0, 1, 0, 0, 1, 0, 0, 0)$ correspondiente con el carácter ASCII $'H'$ del cual partíamos.

Capítulo 5

Criptosistemas de McEliece y Niederreiter

5.1. Criptosistema de McEliece

5.1.1. Introducción

El criptosistema McEliece es un criptosistema de clave pública el cual fue desarrollado por Robert McEliece en 1978. Aunque dicho criptosistema se olvido durante años al tener poca aceptación debido principalmente a los tamaños de las claves que necesita, actualmente ha ganado importancia al ser uno de los criptosistemas resistente a ataques de ordenadores cuánticos.

El criptosistema se basa en la teoría de codificación y su seguridad se encuentra principalmente en que la matriz generatriz del código parece aleatoria y en la dificultad de descodificación de un código lineal del que no conocemos su estructura, al tratarse este de un problema computacionalmente difícil (NP-Hard) cuando el tamaño del código es grande (si es pequeño, encontrar la palabra del código con distancia menor o igual que la enviada no supondría ningún problema).

Para algunos tipos de códigos existen algoritmos de descodificación, los cuales tienen complejidad polinómica y por ello, nos permiten eliminar el error de una palabra del código de forma eficiente. Por ejemplo, los códigos de Reed-Solomon y los códigos de Goppa que hemos visto en los capítulos 3 y 4.

Para el criptosistema de McEliece emplearemos los códigos de Goppa clásicos binarios, debido a que son seguros, sencillos de construir y emplear y presentan mejores propiedades que los códigos de Goppa construidos sobre otros cuerpos finitos según lo visto en la proposición 4.8.

En cuanto a los códigos Reed-Solomon y Reed-Solomon generalizados, no los emplearemos para la construcción del criptosistema de McEliece ya que, como veremos mas adelante, estos se pueden criptoanalizar de forma exitosa.

5.1.2. Descripción del criptosistema

Para realizar la construcción del criptosistema primero escogemos un polinomio separable con coeficientes en $\mathbb{F}_{2^m}$ de grado t y un vector $L \in \mathbb{F}_{2^m}^n$ a partir de los cuales obtenemos un código de Goppa con matriz generatriz, G, de tamaño $k \times n$. Para estos códigos hemos visto el algoritmo de Patterson en las sección 4.2.1, a partir del cual podemos realizar la descodificación de hasta t errores de forma eficiente.

A partir de la matriz generatriz G, calculamos la matriz G', que tiene el mismo tamaño que G y se obtiene mediante el producto de tres matrices: $G' = SGP$ donde S es una matriz no singular cuadrada de tamaño k, es decir, una matriz que es invertible y P es una matriz cuadrada de tamaño n permutacional, es decir, una matriz con todos sus elementos nulos salvo un elemento por cada columna y fila que vale 1.

Analizando la multiplicación de matrices realizada, podemos ver que la matriz SG es también una matriz generatriz del código de Goppa que teníamos, y al multiplicarla por la matriz P tenemos que la matriz G' es la matriz generatriz de un código equivalente al que teníamos, y por ello ambas matrices tienen la misma capacidad correctora. Dicha matriz G' debe caracterizarse por ser una matriz que no revele la estructura del código de Goppa del que partimos, es decir, debe ser una matriz que oculte el código de Goppa inicial.

Entonces las claves pública, κ_p, y privada, κ_s, del criptosistema de McEliece son respectivamente:

$$\kappa_p = (G', t) \text{ y } \kappa_s = (G, S, P, \vartheta)$$

donde ϑ denota el algoritmo de descodificación de hasta t errores del código de Goppa de partida.

5.1.3. Cifrado y descifrado del McEliece

En cuanto al proceso de comunicación entre dos personas mediante el empleo del criptosistema de McEliece, procedemos a explicar tanto el cifrado de un mensaje como su correspondiente descifrado:

- Para realizar el cifrado de un mensaje $m = (m_1, ..., m_k)$, de tamaño k, dirigido a un usuario con clave publica $\kappa_p = (G', t)$, tenemos que multiplicar el mensaje m por la matriz G' y obtenemos el vector c de tamaño n: $c = mG' = (c_1, ..., c_n)$.

Ahora añadimos una cantidad de errores, de tamaño menor o igual a t, al vector c, es decir, tomamos $e = (e_1, ..., e_n)$ con $\omega(e) \leq t$ y realizamos la suma de ambos vectores, $c' = c + e$. Normalmente se escoge un vector de errores cuyo peso sea el máximo.

- Para realizar el descifrado del mensaje $c' = (c'_1, ..., c'_n)$ recibido, podemos observar que como tenemos $c' = c + e = mG' + e$, entonces si multiplicamos dicho vector por la matriz inversa de la matriz de permutación, P^{-1} obtenemos: $c^* = c'P^{-1} = mSG + eP^{-1}$.

  Siempre es posible realizar el paso anterior ya que toda matriz de permutación es invertible.

  Ahora podemos emplear el algoritmo eficiente de descodificación de t errores, ϑ, para obtener mSG. Como el criptosistema que hemos definido emplea códigos de Goppa binarios, dicho algoritmo de descodificación puede ser el visto en la sección 4.2.1. El algoritmo se puede emplear ya que la matriz SG es también una matriz generatriz del código de Goppa y que tanto el vector e como el vector eP^{-1} tienen el mismo peso.

  Después, obtenemos mS resolviendo el sistema de ecuaciones correspondiente a realizar el descifrado de los códigos de Goppa , ya que mS se corresponde con otro posible mensaje.

  Por último, obtenemos el bloque descifrado m a partir de la inversa de la matriz S: $m = mSS^{-1}$.

Ejemplo 5.1. Utilizaremos el código de Goppa que construimos en el ejemplo 4.1. Comenzamos generando las claves pública y privada del criptosistema que emplea dicho código de Goppa. En dicho ejemplo ya mostramos la matriz G de tamaño 4×16, la cual emplearemos para el calculo de la matriz G', necesaria para la clave pública del criptosistema. Para ello, necesitamos obtener tanto una matriz de permutación cuadrada de tamaño 16, P, como una matriz invertible cuadrada de tamaño 4, S.

Para conseguir dichas matrices, emplearemos el código desarrollado en *Sagemath*, el cual se muestra junto con el resto del código generado para la construcción del criptosistema de McEliece en [1, 5.2] del trabajo fin de grado de informática. Las matrices generadas son:

$$S = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix}
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

Entonces obtenemos la matriz G' como el producto de las matrices S, G y P:

$$G' = S \cdot G \cdot P = \begin{pmatrix}
0&0&1&0&0&1&0&0&0&0&1&0&1&1&1&1\\
1&1&0&1&1&0&0&0&1&1&0&1&0&1&0&0\\
1&0&0&0&1&1&1&0&0&1&0&1&1&0&0&1\\
1&1&0&0&0&0&1&1&0&1&1&0&1&1&0&1
\end{pmatrix}$$

Por tanto, ya tenemos todos los elementos necesarios para generar las claves publica y privada, κ_p y κ_s respectivamente. Ahora vamos a realizar tanto el cifrado como el correspondiente descifrado de un mensaje.

Supongamos que una persona desea enviar el carácter $'H'$ a otra persona que tiene como clave pública κ_p. Ya vimos en el ejemplo 4.1, que dicho carácter ASCII se corresponde con el vector binario $m = (0,1,0,0,1,0,0,0)$.

Para realizar el cifrado del vector m primero dividimos el mensaje m en dos bloques de longitud 4 cada uno:

$$m_1 = (0,1,0,0),\ m_2 = (1,0,0,0)$$

Entonces multiplicamos cada bloque por la matriz G' que obtenemos de su clave pública y como resultado tenemos los bloques codificados:

$$c_1 = (1,1,0,1,1,0,0,0,1,1,0,1,0,1,0,0)$$

$$c_2 = (0,0,1,0,0,1,0,0,0,0,1,0,1,1,1,1)$$

Ahora, obtenemos a partir de la clave pública, que la capacidad correctora del código empleado es $t = 3$. Por tanto, para cada bloque codificado introducimos 3 errores.

Para el primer bloque, introducimos los errores en las posiciones 5, 10 y 14 y para el segundo bloque, los introducimos en las posiciones 0, 1 y 13. Entonces, el vector cifrado, c', para el cual ya se han concatenado los dos bloques cifrados se corresponde con:

$$c' = (1,1,0,1,1,1,0,0,1,1,1,1,0,1,1,0,1,1,1,0,0,1,0,0,0,0,1,0,1,0,1,1)$$

Entonces, se envía el mensaje c' a la persona, con quien se quería realizar la comunicación, a la que pertenecía la clave pública.

En cuanto al proceso de descifrado, seguimos también los pasos de la sección 5.1.3. Por tanto, primero tenemos que calcular la matriz inversa de P y después multiplicamos cada bloque de c' de longitud 16 por dicha matriz, obteniendo como resultado:

$$c_1^* = (1,1,1,1,1,1,0,0,1,1,1,1,0,0,1,0)$$
$$c_2^* = (0,0,0,1,1,0,0,1,0,1,0,1,0,1,1,1)$$

Ahora, para ambos bloques tenemos que aplicar el algoritmo de descodificación de Patterson de igual forma que vimos en el ejemplo 4.1. Por tanto, nos limitaremos simplemente a mostrar los resultados obtenidos por el programa realizado en *Sagemath* en [1, 4.3.2] y [1, 4.3.3] del trabajo fin de grado de informática.

Calculamos el síndrome de ambos bloques según lo visto en la definición 4.9:

$$s(c_1^*) = \sum_{i=0}^{15} \frac{c_{1_i}^*}{x - l_i} \equiv (\alpha^3 + \alpha)x^2 + x + \alpha^3 \bmod g(x)$$

$$s(c_2^*) = \sum_{i=0}^{15} \frac{c_{2_i}^*}{x - l_i} \equiv (\alpha^3 + \alpha^2 + 1)x^2 + (\alpha^3 + \alpha^2)x + \alpha^3 \bmod g(x)$$

donde l_i hace referencia a la coordenada i-esima del vector L correspondiente al código de Goppa, $\Gamma(L, g)$, empleado.

Ahora calculamos los polinomios $f_1(x)$ y $f_2(x)$, los cuales obtenemos a partir del inverso del síndrome $s(c_1^*)$ y $s(c_2^*)$ módulo g respectivamente:

$$f_1(x) = \alpha^3 x^2 + (\alpha^3 + \alpha + 1)x + \alpha^2$$

$$f_2(x) = (\alpha^2 + \alpha)x + \alpha^3 + \alpha^2 + 1$$

Como $f_1(x) \neq x$ y $f_2(x) \neq x$ tenemos que continuar con los pasos restantes del algoritmo en ambos casos. Por ello, ahora calculamos $h_i(x)$ de forma que $h_i^2(x) \equiv f_i(x) + x \bmod g(x)$, con $i = 1, 2$:

$$h_1(x) = (\alpha + 1)x^2 + (\alpha^3 + \alpha^2)x + \alpha$$

$$h_2(x) = (\alpha^3 + \alpha^2 + \alpha + 1)x^2 + (\alpha^3 + \alpha^2 + 1)x + \alpha^3 + 1$$

Ahora obtenemos los polinomios $a_i(x)$ y $b_i(x)$ de menor grado para cada polinomio h_i, con $i = 1, 2$:

$$a_1(x) = \alpha^3 x + \alpha^3 + \alpha + 1,\ b_1(x) = (\alpha^3 + \alpha^2 + \alpha)x + \alpha^3 + \alpha^2 + 1)$$

$$a_2(x) = \alpha^2 x + \alpha^3 + \alpha,\ b_2(x) = \alpha^3 x + \alpha + 1$$

Por tanto el polinomio localizador de errores resultante para cada bloque es:

$$L_1(x) = (\alpha^3 + \alpha + 1)x^3 + (\alpha^3 + \alpha^2)x^2 + (\alpha^3 + \alpha^2 + \alpha)x + \alpha^3 + 1$$

$$L_2(x) = (\alpha^3 + \alpha^2)x^3 + (\alpha + 1)x^2 + (\alpha^2 + 1)x + \alpha^3$$

Por último, obtenemos las tres raíces de cada polinomio L_i, con $i = 1, 2$, evaluando sobre cada polinomio cada elemento del vector L.

Para el primer bloque, obtenemos que las raíces son los elementos α^{10}, α^{12} y 1. Por tanto, los errores se encuentran en las posiciones 9, 11 y 14.

En cuanto al segundo bloque, obtenemos que las raíces son los elementos α^3, α^4 y α^5 y por tanto los errores se encuentran en las posiciones 2, 3 y 4.

Entonces, los dos vectores obtenidos una vez corregidos las posiciones de error se corresponden con cada bloque de mensaje multiplicado por la matriz S y codificado, es decir:

$$m_1 \cdot S \cdot G = (1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0)$$

$$m_2 \cdot S \cdot G = (0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1)$$

Realizando la descodificación de cada bloque m_i, obtenemos como resultado los bloques $m_i S$:

$$m_1 \cdot S = (1, 1, 1, 1)$$

$$m_2 \cdot S = (0, 0, 1, 0)$$

Por ultimo, calculando S^{-1}, la matriz inversa de S y después multiplicando cada uno de los bloques $m_i \cdot S$ por S^{-1} recuperamos las coordenadas del mensaje enviado:

$$m_1 = m_1 \cdot S \cdot S^{-1} = (0, 1, 0, 0)$$

$$m_2 = m_2 \cdot S \cdot S^{-1} = (1, 0, 0, 0)$$

Entonces juntando los dos bloques de mensaje, recuperamos el mensaje $m = (0, 1, 0, 0, 1, 0, 0, 0)$ correspondiente con el carácter ASCII $'H'$ que habíamos enviado.

5.1.4. Ventajas y desventajas del criptosistema

El criptosistema de McEliece presenta como una de las principales ventajas, un rápido proceso de codificación y descodificación, los cuales son mas rápidos que los procesos que realizan actualmente muchos de los sistemas mas empleados, como es el caso del sistema RSA.

Otra de sus ventajas es que es un criptosistema post-cuántico, es decir, es resistente a ataques realizados mediante ordenadores cuánticos que emplean el algoritmo de Shor.

El algoritmo de Shor, es un algoritmo cuántico que fue desarrollado en 1994 y permite factorizar un numero entero en factores primos en tiempo polinomial. Por ello, dicho algoritmo nos permitiría romper el criptosistema RSA y el del logaritmo discreto entre otros, los cuales, son los mas empleados actualmente a la hora de realizar cualquier cifrado asimétrico.

A pesar de esto, aunque el criptosistema presenta pocas desventajas, la mas importante es que el tamaño de las claves, tanto pública como privada es muy grande, al ser matrices de elevadas dimensiones.

Por ejemplo, los tamaños de parámetros sugeridos por McEliece fueron $n = 1024$, $k = 524$, $t = 50$, los cuales dan lugar a una clave pública de tamaño cercano a 2^{19} bits.

Sin embargo, debido al avance tecnológico producido en estos años, el tamaño de los parámetros sugerido ha sido elevado a $n = 2048$, $k = 1750$, $t = 27$, de esta forma podemos obtener 80 bits de seguridad, es decir, es necesario realizar 2^{80} operaciones para *"romper"* el criptosistema. Esto permite evitar ataques realizados por fuerza bruta con los ordenadores actuales. Sin embargo, para resistir ataques realizados

mediante ordenadores cuánticos, el tamaño de estos parámetros se debe elevar hasta $n = 6960$, $k = 5400$, $t = 119$, los cuales nos proporcionan un tamaño de clave pública cercano a 2^{23} bits.

Incrementar el tamaño de los parámetros esta relacionado con el incremento de los tamaños de las claves pública y privada empleadas para el criptosistema. Entonces, como estos parámetros se han incrementado notablemente en relación a los sugeridos en la version original, la cual ya presentaba un tamaño de claves alto, tenemos como resultado un tamaño de claves muy elevado.

Otro inconveniente que presenta este criptosistema es que no se puede emplear para producir firmas digitales, aunque este problema ha sido solventado empleando el criptosistema Niederreiter, el cual es un criptosistema dual al criptosistema de McEliece que explicamos a continuación, que permite la realización de firmas digitales [7].

5.2. Criptosistema Niederreiter

5.2.1. Introducción

El criptosistema Niederreiter es un criptosistema dual al de McEliece el cual surgió en 1986, donde se emplea la matriz de control para realizar tanto el cifrado y descifrado. Para dicho criptosistema fueron propuestos en su primera version códigos GRS (Reed-Solomon generalizados).

Mas adelante se descubrió que dicho criptosistema se criptoanalizaba exitosamente si se empleaban códigos Reed-Solomon generalizados, como mostramos mas adelante. Sin embargo, si se emplean en su lugar códigos de Goppa clásicos su seguridad hasta el momento no se ha visto comprometida, de igual forma que para el criptosistema de McEliece.

Esto es debido a que los critosistemas de Niederreiter y McEliece son equivalentes en términos de seguridad, lo cual, fue probado por Yuan Xing en el año 1994 en [6].

5.2.2. Descripción del criptosistema

Para su construcción, se realiza un proceso semejante al realizado en el sistema de McEliece, es decir, se elige un polinomio separable y un vector L, que se emplean para generar un código de Goppa capaz de corregir t errores a partir de un algoritmo de descodificación eficiente, y ahora se calcula la matriz de control del código, H, que tiene un tamaño $(n-k) \times n$. Una vez calculada dicha matriz de control debemos enmascararla, de igual forma que se enmascaraba la matriz generatriz del código para el criptosistema de McEliece como vimos en la sección 5.1.2. Por tanto, se eligen S, una

matriz no singular cuadrada de tamaño $n-k$ y P, una matriz cuadrada de tamaño n de permutación.

Entonces calculamos la matriz $H' = S \cdot H \cdot P$ y la clave pública, κ_p, y la clave privada, κ_s, del criptosistema son respectivamente:

$$\kappa_p = (H', t) \text{ y } \kappa_s = (H, S, P, \vartheta)$$

donde ϑ denota un algoritmo decodificador por sindromes, similar al visto en la sección 2.2, eficiente de hasta t errores del código de Goppa de partida.

5.2.3. Cifrado y descifrado del Niederreiter

En cuanto al proceso de cifrado y descifrado del criptosistema se realiza un procedimiento semejante al realizado en McEliece en la sección 5.1.3.

Para el cifrado se emplea la matriz H' en lugar de la matriz G' y únicamente se pueden cifrar palabras de peso menor o igual a t, las cuales, se corresponden con los lideres de los cogrupos que vimos en la sección 2.2.

Para el descifrado el procedimiento a realizar es el mismo pero empleando en primer lugar la matriz inversa de S y no la de P.

Ejemplo 5.2. Vamos a construir un criptosistema de Niederreiter empleando su propuesta original, es decir, a partir de un código Reed-Solomon generalizado. Por ello, consideramos el código $GRS_3(a)$ sobre $\mathbb{F}_7$, donde $a = (0,1,2,3,4,5,6)$, luego los parámetros fundamentales del código son $[7,3,5]$. Una matriz de control, H, de $GRS_3(a)$ es

$$H = \begin{pmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 0 & 6 & 5 & 4 & 3 & 2 & 1 \\ 0 & 6 & 3 & 5 & 5 & 3 & 6 \\ 0 & 6 & 6 & 1 & 6 & 1 & 1 \end{pmatrix}$$

A partir de la matriz H, construimos el criptosistema de Niederreiter empleando para ello una matriz aleatoria cuadrada invertible, S, de tamaño 4 y una matriz aleatoria de permutación cuadrada, P, de tamaño 7:

$$S = \begin{pmatrix} 4 & 0 & 2 & 2 \\ 0 & 3 & 1 & 0 \\ 1 & 6 & 3 & 2 \\ 0 & 6 & 5 & 5 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A partir de ellas generamos la matriz $H' = S \cdot H \cdot P$ que empleamos como clave pública del criptosistema:

$$H' = \begin{pmatrix} 4 & 3 & 0 & 1 & 6 & 4 & 3 \\ 2 & 2 & 4 & 3 & 3 & 0 & 0 \\ 1 & 4 & 1 & 5 & 2 & 2 & 6 \\ 4 & 6 & 5 & 5 & 5 & 3 & 0 \end{pmatrix}$$

Ahora realizamos el cifrado de un mensaje, m, con peso no superior a la capacidad correctora de $GRS_3(a)$, que es $t = 2$. Tomamos $m = (0, 2, 0, 0, 5, 0, 0)$ y multiplicamos la clave pública por el mensaje obteniendo $c = H' \cdot m^T = (1, 5, 4, 2)$.

En cuanto al proceso de descifrado, multiplicamos en primer lugar por la inversa de la matriz S, $c_1 = S^{-1} \cdot c^T = (0, 4, 0, 4)$ y ahora aplicamos el algoritmo eficiente de descodificación por síndrome obteniendo el vector $m_1 = (0, 5, 0, 0, 0, 0, 2)$. Por último, realizamos la multiplicación de la matriz inversa de P por m_1 para recuperar el mensaje de partida, $m = P^{-1} \cdot m_1^T = (0, 2, 0, 0, 5, 0, 0)$.

5.3. Posibles ataques

Como hemos mencionado en la sección 5.2, por [6] sabemos que los critosistemas de Niederreiter y McEliece son equivalentes en términos de seguridad. Por tanto, consideramos sin perdida de generalidad un criptosistema de McEliece construido a partir del código C de parámetros $[n, k, d]$ y matrices S y P.

Vamos a mostrar muy brevemente los dos posibles ataques que se pueden realizar contra el criptosistema de McEliece, al ser un criptosistema que emplea códigos para su construcción:

- *Ataques genéricos de descodificación*: Se trata de intentar recuperar el mensaje original, m, que hemos cifrado obteniendo el vector y, mediante el empleo del código C' que se obtiene al considerar la matriz pública del criptosistema, G', como matriz generatriz para dicho código.

Para ello, los mejores resultados se obtienen empleando algoritmos que mejoran la descodificación del conjunto de información. Esta descodificación trata de encontrar k coordenadas del vector y que no presenten ningún error. Entonces, tomando dichas coordenadas del vector y y la matriz inversa formada por las k columnas seleccionadas de la matriz G' podemos recuperar el mensaje original m.

Uno de los algoritmos de descodificación del conjunto de información mas conocido es el de *Lee-Brickell* [36].

- *Ataques contra la estructura del código*: En este caso se trata de intentar recuperar, las matrices G, S y P a partir de las cuales se ha realizado la construcción del criptosistema de McEliece empleado. Para este tipo de ataques los criptosistemas de McEliece que emplean códigos de Goppa son los que presentan una elevada seguridad.

  Por otro lado, el ataque por filtración que mostramos en el capítulo 6, es un ejemplo de ataque contra la estructura del código cuando se emplean los códigos Reed-Solomon generalizados, ya que permite recuperar en este caso los parámetros del código C'. También existen otros tipos de ataques contra la estructura del código para otros códigos, como Reed-Muller, BCH, cuasi-cíclicos entre otros.

Capítulo 6

Ataque contra el criptosistema McEliece con códigos GRS

6.1. Introducción

Como vimos en el capítulo 3, los códigos Reed-Solomon y Reed-Solomon generalizados presentan muy buenas propiedades y por tanto son ampliamente usados en teoría de códigos y sus aplicaciones. Debido a esto fueron propuestos en diferentes criptosistemas como el de Niederreiter hasta que fueron criptoanalizados de forma exitosa. El criptoanalisis mas conocido de estos códigos es el llamado ataque de Sidelnikov y Shestakov [8], el cual permite recuperar la estructura de cualquier código Reed-Solomon generalizado en tiempo polinomial, obteniendo para ello el par de vectores a y b que definen el código Reed-Solomon generalizado. Para realizar este ataque es necesario calcular en una primera etapa las palabras del código de peso mínimo.

En su lugar, vamos a explicar otro ataque, llamado ataque por filtración donde no es necesario realizar esa primera etapa. Para dicho ataque, mostramos una forma estándar que nos permite recuperar los vectores a, b de un código GRS_k mediante el conocimiento de la matriz generatriz de GRS_k y de GRS_{k-1}. Después rebatimos una forma de obtener los vectores a y b mostrada en [33] y [34], mediante el conocimiento únicamente de la matriz pública de cifrado, es decir, una matriz generatriz de GRS_k enmascarada, aplicando una modificación sobre el ataque estándar. Por último, mostramos otra modificación del ataque estándar, que se explica en [32], a partir de la cual si seremos capaces de recuperar los vectores a y b empleando solo la matriz generatriz de GRS_k o pública de un criptosistema de McEliece construido a partir de un código Reed-Solomon generalizado.

Este ataque es menos eficiente que el ataque de Sidelnikov y Shestakov desde el punto de vista computacional ya que mientras la complejidad del ataque por filtra-

ción es $O(k^2n^3 + k^3n^2)$, la del ataque de Sidelnikov y Shestakov es $O(k^3 + k^2n)$. Sin embargo, el ataque por filtración, ademas de ser un ataque válido para los códigos Reed-Solomon generalizados, también se puede emplear en otros códigos como los códigos Reed-Muller, o los códigos algebraico-geométricos, en los cuales, la etapa de cálculo de palabras de código de peso mínimo se realiza en tiempo subexponencial y exponencial respectivamente, y por tanto, el ataque de Sidelnikov y Shestakov no sería factible para criptoanalizar dichos códigos.

Por otro lado, otro objetivo de este capítulo es mostrar que aunque el criptosistema de McEliece parezca ser resistente frente a ataques realizados mediante un ordenador cuántico, para ello es necesario emplear códigos que actualmente sigan siendo seguros, ya que si no, podremos criptoanalizarlos exitosamente sin necesidad de emplear ningún ordenador cuántico.

Antes de mostrar los ataques, vamos a introducir un conjunto de definiciones y propiedades esenciales de la teoría de códigos que necesitaremos emplear cuando desarrollemos el ataque por filtración.

Definición 6.1. Sean $x = (x_1, ..., x_n)$ e $y = (y_1, ..., y_n)$ dos elementos de $\mathbb{F}_q^n$. El producto por componentes o producto estrella, denotado por $x \star y$, se define como el producto escalar de las respectivas componentes de ambos vectores:

$$x \star y = (x_1y_1, ..., x_ny_n)$$

La potencia i-esima del vector x, denotado por $x^{\star i}$, será el resultado de realizar el producto estrella i veces, es decir, $x^{\star i} = x \star x \star \cdots \star x$.

Por último, dado un vector $x = (x_1, ..., x_n)$ que cumple que $x_i \neq 0 \, \forall i = 1, ..., n$, definimos elemento inverso del vector x como, $x^{\star -1} = (x_1^{-1}, ..., x_n^{-1})$

Una vez definido el producto por componentes de dos vectores, resulta bastante intuitiva la definición de producto de códigos que mostramos ahora:

Definición 6.2. Dados C_1 y C_2, dos códigos de igual longitud. Diremos que el código de producto estrella obtenido a partir de los códigos C_1 y C_2, denotado por $C_1 \star C_2$, es el espacio vectorial obtenido a partir de todos los productos por componentes donde el primer vector es un elemento de C_1 y el segundo vector es un elemento de C_2:

$$C_1 \star C_2 = \{x \star y \mid x \in C_1 \, , y \in C_2\}$$

Nota 6.3. También, si las dimensiones de C_1 y C_2 son k_1 y k_2 respectivamente y los vectores $a_1, ..., a_{k_1}$ forman una base del código C_1 y los vectores $b_1, ..., b_{k_2}$ forman una base del código C_2, el código de producto estrella de C_1 y C_2 esta generado por todos los productos estrellas de un vector de una base con otro vector de la otra base:

$$C_1 \star C_2 = \langle a_i \star b_j \mid i = 1, ..., k_1 \ , \ j = 1, ..., k_2 \rangle$$

Definición 6.4. Llamaremos código cuadrado de C_1, denotado por $C_1 \star C_1$ o $C_1^{\star 2}$, al código obtenido a partir del producto estrella del mismo código, es decir, considerando en la definición anterior que $C_2 = C_1$.

Proposición 6.5. *Dados C_1 y C_2, dos códigos de igual longitud, se verifica que:*

1) $dim(C_1 \star C_2) \leq dim(C_1)dim(C_2)$

2) $dim(C_1^{\star 2}) \leq \binom{dim(C_1)+1}{2}$

Demostración.

1) Supongamos que las dimensiones de C_1 y C_2 son k_1 y k_2 respectivamente y los vectores $a_1, ..., a_{k_1}$ forman una base del código C_1 y los vectores $b_1, ..., b_{k_2}$ forman una base del código C_2. Por la definición anterior, tenemos que el código de producto estrella esta generado por los productos estrellas de vectores de cada base.

   Por tanto, si todos los vectores obtenidos a partir de los productos estrellas realizados son linealmente independientes, tendremos que todos ellos forman una base del código de producto estrella. Si no todos los vectores obtenidos son linealmente independientes, eliminando los vectores que son dependientes de otros obtendríamos una base del código de producto estrella. Como el total de productos sería k_1k_2 entonces tendríamos la desigualdad.

2) Supongamos que la dimensión de C_1 es k_1 y los vectores $a_1, ..., a_{k_1}$ forman una base del código C_1. Podemos observar que el código cuadrado esta generado por los productos estrella $a_i \star a_j$ donde $1 \leq i \leq j \leq k_1$ debido a que el producto estrella es conmutativo al ser el producto escalar conmutativo. A partir de esto, razonando de forma semejante a la anterior obtenemos la desigualdad.

□

Proposición 6.6. *Sean $x, y, y' \in \mathbb{F}_q^n$, se cumple que:*

1) *Si $k + k' - 1 \leq n$ entonces $GRS_k(x, y) \star GRS_{k'}(x, y') = GRS_{k+k'-1}(x, y \star y')$.*

2) *Si $2k - 1 \leq n$ entonces $GRS_k(x, y)^{\star 2} = GRS_{2k-1}(x, y \star y)$.*

Demostración.

1) $\subseteq$) Sean $a \in GRS_k(x, y)$ y $b \in GRS_{k'}(x, y')$ elementos cualesquiera. Dichos elementos tendrán la forma siguiente:

   $a = (y_1 p(x_1), ..., y_n p(x_n))$ y $b = (y'_1 q(x_1), ..., y'_n q(x_n))$ donde p y q son ambos polinomios de grado menor que k y k' respectivamente.

   Ahora, calculamos el producto estrella de los vectores a y b:

$$a \star b = (y_1 y'_1 p(x_1) q(x_1), ..., y_n y'_n p(x_n) q(x_n)) = (y_1 y'_1 r(x_1), ..., y_n y'_n r(x_n))$$

   donde r es un polinomio de grado menor que $k + k' - 1$ y por tanto es un elemento de $GRS_{k+k'-1}(x, y \star y')$.

   $\supseteq$) Dado un elemento $a \in GRS_{k+k'-1}(x, y \star y')$, tenemos que $a = (y_1 y'_1 r(x_1), ..., y_n y'_n r(x_n))$ siendo r un polinomio de grado menor que $k + k' - 1$. Dicho polinomio se podrá obtener como combinación lineal de productos de dos polinomios de grado menor que k y k' y por tanto el vector a sera el producto estrella de un elemento de $GRS_k(x, y)$ y otro de $GRS_{k'}(x, y')$.

2) Es evidente su demostración a partir de 1) al ser un caso particular tomando $y = y'$ y $k = k'$.

□

Nota 6.7. Si tenemos que la dimension del GRS verifica que $2k - 1 > n$, la segunda parte de la proposición 6.6 también se cumple ya que, considerando su código dual, sabemos por la proposición 3.6 que la dimension sera $k' = n - k$ y entonces como $k > \frac{n+1}{2}$ se tiene que $-k \leq \frac{-n-1}{2}$ y por tanto $k' \leq \frac{n-1}{2} \leq \frac{n+1}{2}$.

Para la primera parte de la proposición 6.6 también se puede realizar un razonamiento similar teniendo en cuenta tanto la dimensión k como k', pero este caso no lo mostramos debido a que no lo necesitaremos emplear después.

Por tanto, como principal conclusión obtenemos que la dimension de un código cuadrado de un GRS de dimension k es $2k - 1$.

Ahora que sabemos la dimension del código cuadrado de un GRS, vamos a introducir una definición y proposición que necesitaremos para demostrar una importante proposición, la cual nos permite conocer con alta probabilidad la dimensión del cuadrado de un código lineal aleatorio.

Definición 6.8. Sea C un código lineal de parámetros $[n, k]$ sobre $\mathbb{F}_q$, el cual tiene como base a los vectores $g_1, .., g_k$. Llamaremos segunda potencia simétrica o producto tensorial simétrico de C con el mismo, y lo denotamos por $S^2(C)$, al código que tiene como

base $\{g_i \otimes g_j \mid 1 \leq i \leq j \leq k\}$, donde $\otimes$ hace referencia al operador producto tensorial. Por tanto, la dimension de $S^2(C)$ es $\binom{k+1}{2}$.

Considerando ahora la aplicación lineal

$$\sigma : S^2(C) \longrightarrow C^{\star 2}$$

$$g_i \otimes g_j \longmapsto g_i \star g_j$$

tenemos por el primer teorema del isomorfismo, al ser $C^{\star 2}$ la imagen de σ, que

$$dim\, K^2(C) + dim\, C^{\star 2} = dim\, S^2(C) = \binom{k+1}{2} \tag{6.1}$$

siendo $K^2(C)$ el núcleo de la aplicación σ, es decir, $K^2(C) = Ker(\sigma)$

Por tanto, tenemos que $K^2(C)$ es el espacio de soluciones del conjunto de ecuaciones dadas por

$$\sum_{1 \leq i \leq i' \leq k} g_{ij} g_{i'j} x_{ii'} = 0 \mid 1 \leq j \leq n$$

Proposición 6.9. *Sea C un código lineal de parámetros $[n,k]$ y sea G una matriz generatriz de C, que tomamos en forma sistemática, es decir, $G = (I_k \mid P)$ siendo P una matriz de tamaño $k \times (n-k)$. Entonces se cumple que*

$$dim\, K(L_P) = dim\, K^2(C^{\perp})$$

donde $K(L_P)$ es el núcleo de L_P, que hace referencia al sistema asociado a la matriz P, formado por k ecuaciones y $\binom{n-k}{2}$ variables, $x_{jj'}$ con $k < j < j' \leq n$, es decir

$$L_P = \left\{ \sum_{k<j<j' \leq n} p_{ij} p_{ij'} x_{jj'} = 0 \mid 1 \leq i \leq k \right\}$$

Demostración. Consideramos G, matriz generatriz de C en forma sistemática, entonces la matriz de control de C es $H = (P^T \mid -I_{n-k})$.

Sea h_i la fila i-esima de la matriz de control H, e_i la i-esima fila de la matriz identidad I_{n-k} y q_i la i-esima fila de la matriz P^T. Entonces se cumple que $q_{ij} = p_{j\,i+k}$ y $h_i = (q_i \mid -e_i)$.

Por tanto $h_j \star h_{j'}$ es $(q_j \star q_j \mid e_i)$ si $j = j'$ y $(q_j \star q_{j'} \mid 0)$ si $j < j'$.

Sea M_1 la matriz de tamaño $k \times \binom{n-k}{2}$ formada por los elementos $p_{ij}p_{ij'}$ con $1 \leq i \leq k$ y $k < j < j' \leq n$. Entonces se tiene que

$$dim\, K(L_P) = \binom{n-k}{2} - rg(M_1)$$

donde $rg(M_1)$ hace referencia al rango de la matriz M_1.

Consideramos ahora M_2, la matriz de tamaño $\binom{n-k+1}{2} \times n$ formada por los elementos $h_{ij}h_{i'j}$ con $1 \leq i \leq i' \leq n-k$ y $1 \leq j \leq n$. Entonces se tiene que

$$dim\, (C^{\perp})^{\star 2} = rg(M_2) = n - k + rg(M_1)$$

Por tanto, tenemos que

$$dim\, K(L_P) = \binom{n-k}{2} + n - k - dim\, (C^{\perp})^{\star 2}$$

y por la igualdad de la ecuación (6.1) tenemos que

$$dim\, K(L_P) = \binom{n-k}{2} + n - k - (\binom{n-k+1}{2} - dim\, K^2(C^{\perp}))$$

y como $\binom{n-k}{2} + n - k = \binom{n-k+1}{2}$ entonces tenemos que $dim\, K(L_P) = dim\, K^2(C^{\perp})$ como queríamos. □

Corolario 6.10. *Sea C un código lineal de parámetros $[n,k]$ sobre el cuerpo $\mathbb{F}_q$. Se verifica que*

$$dim\, C^{\star 2} \leq min\left\{n, \binom{k+1}{2}\right\}$$

Ahora vamos a probar lo que realmente buscábamos, que la igualdad en la desigualdad del corolario 6.10 se da con una alta probabilidad si las entradas de la matriz P son independientes e igualmente distribuidas.

Proposición 6.11. *Sea C un código lineal aleatorio de parámetros $[n,k]$ sobre el cuerpo $\mathbb{F}_q$ tal que $n > \binom{k+1}{2}$. Entonces se verifica que*

$$Pr\left(dim(C^{\star 2}) = \binom{k+1}{2}\right) = 1$$

donde $Pr(S)$ indica la probabilidad de que ocurra el suceso S.

Demostración. Sea C un código lineal que verifica las hipótesis de la proposición. Si G es una matriz generatriz de C, la cual tomamos en forma sistemática, es decir, $G = [I_k \mid P]$, donde I_k hace referencia a la matriz identidad cuadrada de tamaño k y P al resto de la matriz, la cual tiene tamaño $k \times (n-k)$. Sabemos que $H = [P^T \mid -I_{n-k}]$ es una matriz de control de C y por tanto, también es una matriz generatriz de $C^{\perp}$. Si denotamos por L_{P^T} a el sistema lineal obtenido a partir de la matriz P^T

$$L_{P^T} = \left\{ \sum_{1 \leq j < j' < k} p_{ij} p_{ij'} x_{jj'} = 0 \mid 1 \leq i \leq n-k \right\}$$

el cual tiene $n-k$ ecuaciones lineales y $\binom{k}{2}$ incógnitas, donde p_{ij} hace referencia a el elemento de la fila i y columna j de la matriz P^T y $x_{jj'}$ a una incógnita. La dimensión del espacio de soluciones de L_{P^T} es 0 con una alta probabilidad debido a un resultado probado por Faugère en [9]. Por ello, empleando la ecuación (6.1) tenemos con una alta probabilidad que $dim(C^{\star 2}) = \binom{k+1}{2}$ como queríamos.

□

Debido a esto, tenemos que la dimensión del código cuadrado de un código aleatorio lineal debe encontrarse en torno al $min\{\binom{k+1}{2}, n\}$. Por tanto, esta técnica nos permite diferenciar un código GRS de un código lineal aleatorio, lo cual es muy útil ya que, si por ejemplo, tenemos la clave pública de un criptosistema de McEliece, podremos conocer si dicho criptosistema esta empleando códigos de GRS y por tanto aplicar el ataque por filtración que veremos mas adelante para criptoanalizar dicho criptosistema de McEliece.

Ahora vamos a introducir los códigos punteados y recortados de códigos lineales debido a una propiedad que cumplen los códigos GRS, sobre la cual también nos apoyaremos para realizar el ataque de filtración.

Definición 6.12. Dado un código lineal C de parámetros $[n,k]$, y sea (J, J') una partición de $\{1,...,n\}$ la cual verifica que $J \cup J' = \{1,...,n\}$ y que $J \cap J' = \emptyset$. Diremos que el código punteado de C en J, denotado por $P_J(C)$, es el código, también lineal, cuyas palabras se obtienen a partir de las palabras del código de C restringidas a las posiciones de J':

$$P_J(C) = \{(c_i)_{i \in J'} \mid c \in C\}$$

Nota 6.13. Si G es una matriz generatriz para C, una matriz generatriz para $P_J(C)$ se obtiene eliminando de la matriz G el conjunto de J columnas y omitiendo, en caso de que ocurra, las filas con todos los elementos nulos, filas duplicadas y linealmente dependientes.

Ejemplo 6.1. Sea C un código lineal de parámetros $[7, 3, 3]$ sobre $\mathbb{F}_2$ dado por la matriz generatriz

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

El código punteado en $J = \{5\}$ es el código lineal formado por las palabras del código que se obtienen de eliminar la quinta coordenada de las palabras del código C, y la matriz generatriz es

$$G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Definición 6.14. Dado un código lineal C de parámetros $[n, k]$, y sea (J, J') una partición de $\{1, ..., n\}$ la cual verifica que $J \cup J' = \{1, ..., n\}$ y que $J \cap J' = \emptyset$. Diremos que el código recortado de C en J, denotado por $S_J(C)$, es el código cuyas palabras se obtienen a partir de las palabras del código de C que verifican que sus coordenadas son nulas en las posiciones de J, restringidas a las posiciones de J':

$$S_J(C) = \{(c_i)_{i \in J'} \mid c \in C \,,\, c_j = 0 \, \forall j \in J\}$$

Nota 6.15. Los códigos recortados son un subconjunto de los códigos punteados.

Nota 6.16. Si una matriz generatriz de C es G, una matriz generatriz para $S_J(C)$ se obtiene moviendo las columnas de las posiciones de J a las $|J|$ primeras columnas de la matriz y aplicando después eliminación Gaussiana sobre estas para obtener la matriz identidad en las $|J|$ primeras filas y ceros en la submatriz $k - |J| \times |J|$, es decir, obteniendo

$$G = \begin{pmatrix} I_{|J|} & T \\ O & G' \end{pmatrix}$$

donde $I_{|J|}$ es la matriz identidad cuadrada de tamaño $|J|$, T es una matriz cualquiera, O es una matriz con todas sus coordenadas nulas de tamaño $k - |J| \times |J|$ y G' es la matriz generatriz del código $S_J(C)$.

Nota 6.17. Si el conjunto J se reduce únicamente a un valor, es decir, $J = \{j\}$, escribiremos $P_j(C)$ y $S_j(C)$ en lugar de escribir $P_{\{j\}}(C)$ y $S_{\{j\}}(C)$ respectivamente.

Ejemplo 6.2. Sea C un código lineal de parámetros $[7, 3, 3]$ sobre $\mathbb{F}_2$ dado por la misma matriz generatriz que el ejemplo 6.1. El código recortado en la quinta coordenada, $S_5(C)$, es el código formado por las palabras del código que se obtienen de eliminar la quinta coordenada de las palabras del código C y que su quinta coordenada sea nula.

Moviendo la quinta columna hacia la primera y realizando eliminación Gaussiana obtenemos

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

y por tanto, obtenemos que la matriz generatriz de $S_5(C)$ es

$$G' = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Proposición 6.18. *Dado C un código $GRS_k(a, b)$, se cumple que el código recortado de C es también un código GRS y se tiene que:*

$$S_J(C) = GRS_{k-|J|}(P_J(a), b')$$

donde las coordenadas de b' vienen dadas por:

$$b'_i = b_i \prod_{j \in J} (a_i - a_j)$$

Demostración. Sea J el conjunto de posiciones que emplearemos para el código recortado y G la matriz generatriz de C dada por

$$G = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ b_1 a_1 & b_2 a_2 & \cdots & b_n a_n \\ \vdots & \vdots & \ddots & \vdots \\ b_1 a_1^{k-1} & b_2 a_2^{k-1} & \cdots & b_n a_n^{k-1} \end{pmatrix}$$

Moviendo las columnas de las posiciones de J a las $|J|$ primeras columnas de la matriz y aplicando después eliminación Gaussiana obtenemos la matriz

$$G = \begin{pmatrix} I_{|J|} & T \\ O & G' \end{pmatrix}$$

donde G' es la matriz generatriz de $S_J(C)$, la cual tiene la forma

$$G' = \begin{pmatrix} b'_{|J|+1} & \cdots & b'_n \\ \vdots & \ddots & \vdots \\ b'_{|J|+1} a_{|J|+1}^{k-|J|-1} & \cdots & b'_n a_n^{k-|J|-1} \end{pmatrix}$$

siendo $b'_i = b_i \prod_{j \in J} (a_i - a_j)$ debido a las operaciones necesarias para obtener la matriz identidad, $I_{|J|}$ y la matriz cuyos elementos son todos nulos, O. Ademas, vemos que al ser $b'_i \in \mathbb{F}_q \; \forall i = 2, ..., n$, se tiene que G' es una matriz también de un código GRS. □

6.2. Ataque por filtración a códigos GRS

6.2.1. Ataque estándar

En un principio, para el ataque estándar vamos a considerar que conocemos los códigos Reed-Solomon generalizados de dimensión k y $k-1$, $GRS_k(a,b)$ y $GRS_{k-1}(a,b)$ respectivamente, donde a y b son elementos de $\mathbb{F}_q^n$ y veamos que si disponemos de estos códigos, es posible obtener el código Reed-Solomon generalizado de dimensión $k-2$ construido a partir de los vectores a y b, lo cual es un paso clave para nuestro ataque.

Proposición 6.19. *Dados los códigos $GRS_k(a,b)$ y $GRS_{k-1}(a,b)$, es posible calcular el código $GRS_{k-2}(a,b)$.*

Demostración. Sabemos por la proposición 6.6 que

$$GRS_{k-2}(a,b) \star GRS_k(a,b) = GRS_{k-1}(a,b)^{\star 2} \tag{6.2}$$

y por tanto, si vemos que dicho código de Reed-Solomon generalizado es

$$GRS_{k-2}(a,b) = \{c \in GRS_{k-1}(a,b) \mid c \star GRS_k(a,b) \subseteq GRS_{k-1}(a,b)^{\star 2}\}$$

estaría probado. Para ello, veamos la doble contención:

$\subseteq$) Sea $c \in GRS_{k-2}(a,b)$, como $GRS_{k-2}(a,b) \subset GRS_{k-1}(a,b)$ entonces $c \in GRS_{k-1}(a,b)$ y por (6.2) la contención esta demostrada.

$\supseteq$) Sea $c \in GRS_{k-1}(a,b)$, entonces puede ocurrir que:

1) $c \in GRS_{k-2}(a,b)$.

2) $c \in GRS_{k-1}(a,b) \setminus c \in GRS_{k-2}(a,b)$. En este caso, existe un polinomio f de grado $k-2$ tal que $c = (b_1 f(a_1), ..., b_n f(a_n))\ \forall c$. Entonces el código obtenido a partir de $c \star GRS_k(a,b)$ esta formado por polinomios cuyo grado es hasta $k-2+k-1 = 2k-3$, luego es un código con dimensión $2k-2$, el cual tiene que estar contenido por hipótesis, según la proposición 6.6, en un código de dimensión $2k-3$, lo cual es absurdo y por tanto este caso no puede ocurrir.

Entonces, tendríamos que solo puede ocurrir que $c \in GRS_{k-2}(a,b)$ que es lo que queríamos. □

Nota 6.20. Para la obtención de forma práctica del código $GRS_{k-2}(a,b)$, tenemos que calcular una base del código $(GRS_{k-1}(a,b)^{\star 2})^{\perp}$ y después resolver el sistema construido a partir de las ecuaciones que tienen la forma $g_i \star h_j$, donde g_i hace referencia a

la fila i-esima de una matriz generatriz de $GRS_k(a,b)$ y h_j hace referencia a la fila j-esima de una matriz de control, H, de $GRS_{k-1}(a,b)^{\star 2}$. Esto es debido a que se cumple en primer lugar que $(c \star g_i) \in GRS_{k-1}(a,b)^{\star 2}$ si y solo si $(c \star g_i) \cdot H^T = 0$, es decir, $(c \star g_i) \cdot h_j = 0$, y en segundo lugar, se verifica que $(c \star g_i) \cdot h_j = c \cdot (g_i \star h_j)$.

Por tanto, a partir de la proposición 6.19, podemos reiterar el proceso con los dos códigos Reed-Solomon generalizados que tengamos de dimensión mas baja hasta obtener el código $GRS_1(a,b)$. Como $GRS_1(a,b) = \{\lambda b \mid \lambda \in \mathbb{F}_q\} = \langle b \rangle$ debido a que sus elementos se obtienen evaluando en constantes y multiplicándolas por el elemento b, a partir de este proceso, conocido como proceso de filtración, podemos obtener el elemento b.

Si ahora obtenemos el elemento a, tendríamos determinado por completo el código Reed-Solomon y por tanto un ataque exitoso contra dicho código. Para la obtención del elemento a, vamos a emplear el código Reed-Solomon de dimension 2, que ya tenemos calculado por la filtración realizada.

A partir de este código, calculamos una matriz generatriz para él, donde n hace referencia al tamaño de los elementos codificados, que es conocido:

$$G = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \end{pmatrix}$$

Como sabemos por la sección 3.2 otra matriz generatriz de $GRS_2(a,b)$ es

$$G' = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ a_1 b_1 & a_2 b_2 & \cdots & a_n b_n \end{pmatrix}$$

Entonces, como la matriz generatriz en forma estándar es única, ambas matrices generatrices del mismo código deben ser equivalentes a la misma matriz generatriz en forma estándar, por lo visto en la proposición 2.10. Por tanto, realizando eliminación de Gauss-Jordan sobre ambas matrices, tenemos que $rref(G) = rref(G')$, donde $rref(A)$ hace referencia a la forma escalonada reducida de la matriz A.

Conocemos completamente $rref(G)$. Ademas, $rref(G')$ tiene la siguiente forma:

$$G' = \begin{pmatrix} 1 & 0 & \frac{(a_2-a_3)b_3}{(a_2-a_1)b1} & \cdots & \frac{(a_2-a_n)b_n}{(a_2-a_1)b1} \\ 0 & 1 & \frac{(a_3-a_1)b_3}{(a_2-a_1)b_2} & \cdots & \frac{(a_n-a_1)b_n}{(a_2-a_1)b_2} \end{pmatrix}$$

donde el elemento b ya lo tenemos calculado. Por tanto igualando ambas matrices, obtenemos un sistema de $2n-4$ ecuaciones lineales con n incógnitas, donde al menos $n-2$ ecuaciones serán linealmente independientes. Además, como sabemos que en un código Reed-Solomon generalizado siempre se pueden fijar tres puntos del vector

a según [27], entonces se puede resolver dicho sistema y por tanto obtener un único vector que se corresponde con el vector a que nos quedaba por calcular.

Ejemplo 6.3. Sean $GRS_4(a,b)$ y $GRS_3(a,2b)$, con valores $a = (1,2,9,3,10,8,5,4) \in \mathbb{F}_{11}$ y $b = (3,10,5,9,9,10,5,2) \in \mathbb{F}_{11}$ que suponemos desconocidos, los códigos Reed-Solomon generalizados de los cuales conocemos una matriz generatriz que denotamos G y G' respectivamente, cuyos valores son

$$G = \begin{pmatrix} 3 & 10 & 5 & 9 & 9 & 10 & 5 & 2 \\ 3 & 9 & 1 & 5 & 2 & 3 & 3 & 8 \\ 3 & 7 & 9 & 4 & 9 & 2 & 4 & 10 \\ 3 & 3 & 4 & 1 & 2 & 5 & 9 & 7 \end{pmatrix} \quad G' = \begin{pmatrix} 6 & 9 & 10 & 7 & 7 & 9 & 10 & 4 \\ 6 & 7 & 2 & 10 & 4 & 6 & 6 & 5 \\ 6 & 3 & 7 & 8 & 7 & 4 & 8 & 9 \end{pmatrix}$$

Como $GRS_3(a,2b) = GRS_3(a,b)$ podemos aplicar la proposición 6.19 para comenzar a realizar la filtración. En primer lugar, calculamos $GRS_2(a,b)$ y para ello es necesario realizar el cálculo del código $GRS_3(a,b)^{\star 2}$ que según la nota 6.3, nos podemos restringir a construir el espacio generado a partir de todos los productos de las filas de la matriz G'. Como tiene 3 filas, se necesita realizar un total de 6 multiplicaciones:

$$(6,9,10,7,7,9,10,4) \star (6,9,10,7,7,9,10,4) = (3,4,1,5,5,4,1,5)$$

$$(6,9,10,7,7,9,10,4) \star (6,7,2,10,4,6,6,5) = (3,8,9,4,6,10,5,9)$$

$$(6,9,10,7,7,9,10,4) \star (6,3,7,8,7,4,8,9) = (3,5,4,1,5,3,3,3)$$

$$(6,7,2,10,4,6,6,5) \star (6,7,2,10,4,6,6,5) = (3,5,4,1,5,3,3,3)$$

$$(6,7,2,10,4,6,6,5) \star (6,3,7,8,7,4,8,9) = (3,10,3,3,6,2,4,1)$$

$$(6,3,7,8,7,4,8,9) \star (6,3,7,8,7,4,8,9) = (3,9,5,9,5,5,9,4)$$

A partir de los 6 vectores obtenidos, tenemos que quedarnos solo con una combinación linealmente independiente para calcular una base, la cual se tiene que obtener a partir de 5 vectores según la proposición 6.6, y como el tercer y cuarto vector son iguales ya tenemos los 5 vectores que forman una base y por tanto, que se corresponden con las filas de la matriz generatriz del código $GRS_3(a,b)^{\star 2}$.

A partir de esto, vamos a obtener una matriz generatriz de $GRS_2(a,b)$. Para ello realizaremos los pasos de la nota 6.20, es decir, calculamos una matriz de control, H_1, de $GRS_3(a,b)^{\star 2}$

$$H_1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 9 & 3 & 4 & 6 \\ 0 & 1 & 0 & 4 & 1 & 7 & 7 & 7 \\ 0 & 0 & 1 & 1 & 10 & 8 & 1 & 2 \end{pmatrix}$$

y ahora tenemos que resolver el sistema formado por $3 * 4 = 12$ ecuaciones obtenidas tras realizar el producto estrella de cada fila de G con cada fila de H_1. Si denotamos $c = (c_1, ..., c_8)$ como la incógnita que tenemos que obtener, el sistema se corresponde con $A \cdot c^T = 0$ siendo A la matriz 12×8 siguiente:

$$A = \begin{pmatrix} 3 & 0 & 0 & 9 & 4 & 8 & 9 & 1 \\ 0 & 10 & 0 & 3 & 9 & 4 & 2 & 3 \\ 0 & 0 & 5 & 9 & 2 & 3 & 5 & 4 \\ 3 & 0 & 0 & 5 & 7 & 9 & 1 & 4 \\ 0 & 9 & 0 & 9 & 2 & 10 & 10 & 1 \\ 0 & 0 & 1 & 5 & 9 & 2 & 3 & 5 \\ 3 & 0 & 0 & 4 & 4 & 6 & 5 & 5 \\ 0 & 7 & 0 & 5 & 9 & 3 & 6 & 4 \\ 0 & 0 & 9 & 4 & 2 & 5 & 4 & 9 \\ 3 & 0 & 0 & 1 & 7 & 4 & 3 & 9 \\ 0 & 3 & 0 & 4 & 2 & 2 & 8 & 5 \\ 0 & 0 & 4 & 1 & 9 & 7 & 9 & 3 \end{pmatrix}$$

Lo resolvemos empleando *Sage*, obteniendo como resultado el espacio vectorial que tiene como base los vectores $(1,0,3,8,9,2,6,6)$ y $(0,1,4,4,7,7,2,5)$, los cuales se corresponden con las filas de una matriz generatriz, G''', de $GRS_2(a,b)$ que necesitábamos.

Ahora a partir de $GRS_3(a,b)$ y $GRS_2(a,b)$ tenemos que realizar el mismo procedimiento, es decir, comenzamos calculando $GRS_2(a,b)^{\star 2}$ donde tenemos que realizar 3 multiplicaciones:

$$(1,0,3,8,9,2,6,6) \star (1,0,3,8,9,2,6,6) = (1,0,9,9,4,4,3,3)$$

$$(1,0,3,8,9,2,6,6) \star (0,1,4,4,7,7,2,5) = (0,0,1,10,8,3,1,8)$$

$$(0,1,4,4,7,7,2,5) \star (0,1,4,4,7,7,2,5) = (0,1,5,5,5,5,4,3)$$

Claramente, los tres vectores son linealmente independientes y por tanto constituyen una base que se corresponde con las filas de una matriz generatriz de $GRS_2(a,b)^{\star 2}$.

Ahora tenemos que resolver de nuevo otro sistema formado por 15 ecuaciones que se obtienen de multiplicar cada fila de una matriz generatriz de $GRS_3(a,b)$ con cada fila de una matriz de control de $GRS_2(a,b)^{\star 2}$. Resolviendo de nuevo el sistema a partir de *Sage* obtenemos como resultado el espacio vectorial construido únicamente a partir del vector $(1,7,9,3,3,7,9,8)$, el cual se corresponde con el vector λb, siendo $\lambda \in \mathbb{F}_{11}$.

Por último para recuperar el vector a, tenemos que calcular la matriz en forma escalonada reducida de G''', la cual ya se encuentra en dicha forma, y ahora resolver el sistema expuesto en la sección 6.2.1, formado en este caso por 12 ecuaciones, para el cual fijamos las dos primeras coordenadas del vector a, $a_1 = 1$ y $a_2 = 2$. Por tanto, tenemos que resolver el sistema $B \cdot a^T = 0$ siendo a el vector que queremos obtener, del cual conocemos las dos primeras coordenadas y B la matriz con la forma:

$$B = \begin{pmatrix} 3 & 6 & 2 & 0 & 0 & 0 & 0 & 0 \\ 8 & 6 & 0 & 8 & 0 & 0 & 0 & 0 \\ 9 & 5 & 0 & 0 & 8 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 4 & 0 & 0 \\ 6 & 3 & 0 & 0 & 0 & 0 & 2 & 0 \\ 6 & 2 & 0 & 0 & 0 & 0 & 0 & 3 \\ 8 & 5 & 9 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 3 & 0 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 & 3 & 0 & 0 & 0 \\ 9 & 6 & 0 & 0 & 0 & 7 & 0 & 0 \\ 5 & 8 & 0 & 0 & 0 & 0 & 9 & 0 \\ 5 & 9 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix}$$

Podemos observar, que el sistema siempre tendrá elementos no nulos en las mismas posiciones y como fijamos además las dos primeras coordenadas de a y sabemos que el sistema siempre tiene que tener solución, podemos restringirnos a calcular una matriz inversa de la matriz cuadrada

$$BB = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

la cual siempre será diagonal, lo que se facilita los cálculos. Dicha matriz inversa la tendremos que multiplicar por el vector de longitud 6, donde los valores de sus coordenadas se obtienen tras calcular $-B[i,1] - B[i,2] * 2$ con $i = 0,1,...,6$. En nuestro caso dicho vector es $(7,2,3,10,10,1)$.

A partir de su resolución, obtenemos los valores $a_3 = 9$, $a_4 = 3$, $a_5 = 10$, $a_6 = 8$, $a_7 = 5$ y $a_8 = 4$, recuperando de esta forma el vector a completamente.

Si obtenemos la matriz generatriz en forma estándar tanto del código con los vectores a y b de partida como los obtenidos, obtenemos la misma matriz:

$$G_e = \begin{pmatrix} 1 & 0 & 0 & 0 & 6 & 9 & 8 & 5 \\ 0 & 1 & 0 & 0 & 7 & 6 & 4 & 9 \\ 0 & 0 & 1 & 0 & 6 & 4 & 4 & 7 \\ 0 & 0 & 0 & 1 & 5 & 10 & 1 & 3 \end{pmatrix}$$

Por tanto, el código que obtenemos de ambas formas es el mismo.

6.2.2. Ataque incompleto al criptosistema de McEliece

A pesar de que el ataque general descrito en la sección 6.2.1 es efectivo, es necesario conocer los códigos GRS de dimensiones k y $k-1$, lo cual, no es habitual ya que por ejemplo, si partimos de un criptosistema de McEliece que emplea códigos de Reed-Solomon generalizados, la clave pública es una matriz generatriz enmascarada del código de dimensión k, y por tanto, el código de dimensión $k-1$ es desconocido.

En [33] y [34] muestran un método que supuestamente permite recuperar los vectores a y b de $GRS_k(a,b)$, a partir de la matriz generatriz de $GRS_k(a,b)$ o de la matriz pública de un criptosistema de McEliece construido a partir del código $GRS_k(a,b)$, el cual no funciona.

Para aplicar dicho método se emplea la proposición 6.18, que nos proporciona una forma de obtener códigos Reed-Solomon de una determinada dimensión a partir de un código Reed-Solomon dado. Mas concretamente, si G denota la matriz generatriz de $GRS_k(a,b)$ o matriz pública de un criptosistema de McEliece construida a partir del código $GRS_k(a,b)$, calculamos la matriz generatriz correspondiente al código recortado en la posición 1 del código con matriz generatriz G, que es $S_1(GRS_k(a,b)) = GRS_{k-1}(P_1(a), b')$, donde $P_1(a)$ hace referencia en este caso al vector a punteado en la primera posición y $b' = (b'_2, ..., b'_n)$ con $b'_j = b_j(a_j - a_1)$.

Por otro lado calculamos el código punteado en la primera posición del código Reed-Solomon generalizado, y obtenemos $P_1(GRS_k(a,b)) = GRS_k(P_1(a), P_1(b))$

Por tanto, ahora tenemos los códigos $GRS_k(P_1(a), P_1(b))$ y $GRS_{k-1}(P_1(a), b')$ y según el método, aplicando las matrices generatrices de dichos códigos podemos construir la filtración para realizar el mismo procedimiento que el empleado en el ataque estándar. Podemos observar que dicho proceso no funciona, ya que para realizar el ataque estándar es necesario emplear dos códigos GRS de dimensiones k y $k-1$

construidos a partir de los vectores a y b idénticos u otros vectores que nos proporcionen el mismo código, por ejemplo, sabemos que $GRS_k(a,b) = GRS_k(a,\lambda b)$. Sin embargo, en este caso tenemos que $GRS_{k-1}(P_1(a),b')$ no genera el mismo código que $GRS_{k-1}(P_1(a),P_1(b))$, ya que $(a_j - a_1)$ varía según lo hace j y por tanto no es posible aplicar el ataque estándar de la sección 6.2.1.

6.2.3. Ataque correcto al criptosistema de McEliece

Ahora vamos a explicar un ataque de filtración que funciona correctamente, el cual fue mostrado en el año 2014 en [32].

Para ello, partimos del código $GRS_k(a,b)$ y suponemos que la dimensión del código es menor o igual que la mitad de la longitud del código, n, es decir, $k \leq \frac{n}{2}$. Si esto no ocurre, por la proposición 3.6 podemos aplicar el procedimiento sobre el código dual, que si cumplirá nuestra restricción y después recuperar, a partir de los del dual, los vectores a y b de partida.

Ahora fijamos las dos primeras coordenadas del vector a con valores 0 y 1 respectivamente, lo cual es posible ya que por [27] sabemos que se pueden fijar 3 puntos del vector a.

Debido a esto, todos los polinomios con 0 como raíz tendrán una codificación con primera coordenada nula. De igual forma ocurre con los polinomios que tengan a 1 como raíz, cuya codificación tendrá la segunda coordenada nula.

Entonces, si construimos un subcodigo de $GRS_k(a,b)$ formado únicamente a partir de las codificaciones de polinomios con una raíz de cierta multiplicidad en 0, dicho subcodigo se corresponde con el código recortado en la primera posición a partir de $GRS_k(a,b)$. De igual forma ocurre con la raíz en 1.

Denotamos por $C(i,j)$ el subcodigo de $GRS_k(a,b)$, con $i,j > 0$ y $i+j \leq k-1$ formado solamente a partir de las codificaciones de polinomios, $f(x)$, que tengan a 0 como raíz con multiplicidad al menos i y a 1 como raíz con multiplicidad al menos j, es decir, $x^i(x-1)^j | f(x)$.

Por tanto, se cumple que $C(1,0) = S_1(GRS_k(a,b))$, $C(0,1) = S_2(GRS_k(a,b))$, $C(1,1) = S_{\{1,2\}}(GRS_k(a,b))$ y denotamos $C(0,0) = GRS_k(a,b)$. A partir de esto, vamos a probar una proposición muy parecida a la proposición 6.19, que nos permitía realizar el ataque estándar, la cual también nos proporciona el método clave para comenzar el ataque por filtración.

Proposición 6.21. *Sea $k \leq \frac{n}{2}$ y el código $GRS_k(a,b)$, entonces $\forall i,j > 0$ tales que $i+j \leq k-2$ se verifica que*

$$C(i+1,j) \star C(i-1,j) = C(i,j)^{\star 2} \tag{6.3}$$

Demostración. Sea $V_{ij} = x^i \cdot (x-1)^j \cdot L_{k-i-j}$, donde L_{k-i-j} denota, al igual que en la definición 3.1, el espacio vectorial formado por todos los polinomios con coeficientes sobre el cuerpo empleado cuyo grado es menor que $k-i-j$. Como L_{k-i-j} tiene dimensión $k-i-j$ entonces V_{ij} también y por tanto el código $C(i,j) = \{b \star f(a) \mid f \in V_{ij}\}$ también.

Por último, finalizamos la demostración al cumplirse que

$$V_{ij}^2 = x^{2i} \cdot (x-1)^{2j} \cdot L_{2k-2i-2j-1} = V_{(i-1)j} \cdot V_{(i+1)j}$$

□

Nota 6.22. Todos los códigos $C(i,j)$ son Reed-Solomon generalizados ya que se obtienen mediante la proposición 6.21 a partir de los códigos $C(1,0)$, $C(0,1)$, $C(1,1)$ y $C(0,0)$, los cuales sabemos que son Reed-Solomon generalizados y por tanto, podemos observar que con las ecuaciones (6.2) y (6.3) estamos calculando los mismos códigos.

6.2.3.1. Ataque

Una vez mostrada la proposición 6.21 y la notación empleada, vamos a explicar el ataque.

1) En primer lugar, empleamos los códigos $C(0,0)$ y $C(1,0)$ para obtener, aplicando la ecuación (6.3) $k-2$ veces, el código $C(k-1,0)$, cuya dimensión es 1 y por tanto su matriz generatriz, $G_{(k-1)0}$, se corresponde solamente con un vector. Obviamente, tenemos que el código $C(k-1,0)$ esta formado mediante la codificación de polinomios de la forma λx^{k-1}, ya que por definición solo se codifican polinomios que tengan en 0 una raíz de multiplicidad al menos $k-1$ y como la dimensión es k el grado de los polinomios no puede ser superior a $k-1$. Entonces, si denotamos $c = G_{(k-1)0}$, tenemos que $c = \lambda(a^{k-1} \star b)$.

2) Por otro lado, calculamos el código $C(k-2,1)$ de nuevo aplicando la ecuación (6.3) $k-3$ veces a partir de los códigos $C(0,1)$ y $C(1,1)$. También el código $C(k-2,1)$ tiene dimensión 1 y denotando $c' = G_{(k-2)1}$, siendo $G_{(k-2)1}$ su matriz generatriz, tenemos que el vector c' es de la forma $\alpha a^{k-2} \star (a-1) \star b$.

3) Entonces, el vector c tiene únicamente su primera coordenada nula y el vector c' tiene solo las dos primeras coordenadas nulas. Por tanto, esta bien definida la división $\frac{c'}{c}$ para todas las coordenadas excepto las dos primeras, la cual se corresponde con $\frac{\beta(a-1)}{a}$. Ahora, como solo hemos fijado las dos primeras coordenadas del vector a y por [27] sabemos que se pueden fijar 3 elementos, podemos seleccionar un valor arbitrario para β que denotamos por β_0. Entonces tenemos que

la aplicación

$$f : \mathbb{F}_q \setminus \{0, 1\} \longrightarrow \mathbb{F}_q$$

$$x \longmapsto \frac{\beta_0(x-1)}{x}$$

es una biyección que representa las distintas coordenadas obtenidas a partir del cociente de c y c' mediante la evaluación de la correspondiente coordenada de a. Por tanto, si calculamos la función inversa

$$f^{-1} : \mathbb{F}_q \setminus \{\beta_0\} \longrightarrow \mathbb{F}_q$$

$$y \longmapsto \frac{\beta_0}{\beta_0 - y} = \frac{1}{1 - \frac{1}{\beta_0} y}$$

y evaluamos en las coordenadas de $\frac{c'}{c}$ podemos recuperar todas las coordenadas del vector a excepto las dos primeras, lo cual no supone ningún problema ya que las hemos fijado al comienzo del ataque.

Hay que tener en cuenta que puede ocurrir que alguna coordenada de $\frac{c'}{c}$ tenga el valor β_0 y por tanto no se podría aplicar el paso anterior. Sin embargo, como nosotros elegimos el valor de β_0, siempre podremos seleccionar un cierto valor de β_0 de forma que la función f^{-1} este bien definida realizando su evaluación en todas las coordenadas de $\frac{c'}{c}$. Esto se debe a que todos las coordenadas del vector a deben ser distintas y por tanto también lo deben ser las de $\frac{c'}{c}$, luego existirá al menos un elemento del cuerpo empleado que no sea una coordenada de $\frac{c'}{c}$ y por tanto, dicho elemento sería un valor posible para β.

4) Para finalizar, una vez que tenemos el vector a podemos recuperar fácilmente el vector b, ya que sabemos que $c = \lambda(a^{k-1} \star b)$. Por tanto, si realizamos la división por coordenadas del vector c entre a^{k-1} recuperamos todas las coordenadas de b excepto la primera, ya que $a_1 = 0$ y por tanto la división no esta definida. Entonces para recuperar la coordenada restante, podemos ir asignándola distintos elementos del cuerpo $\mathbb{F}_q$ empleado hasta obtener un vector que sea una palabra del código $GRS_k(a, b)$ de partida.

Por tanto, ahora que ya conocemos los vectores a y b del código empleado, también disponemos del algoritmo de corrección de errores del código. Entonces, si una persona realiza el cifrado de un mensaje m obteniendo y, nosotros aplicamos el algoritmo de corrección de errores que conocemos y nos permite recuperar c, ahora obteniendo una matriz cuadrada invertible de la matriz pública de partida, podemos recuperar el mensaje m de partida.

Nota 6.23. Destacamos que en el paso 3 el valor de β no siempre puede ser 1, como sugiere el artículo [32], ya que puede ocurrir que el vector $\frac{c'}{c}$ tenga alguna coordenada con valor 1 y entonces la función f^{-1} no estaría bien definida. Por ejemplo, si realizamos el ataque de filtración sobre el código $GRS_4(a,b)$ construido en $\mathbb{F}_{17}$, con $a = (0,1,2,3,4,5,6,7,8,9)$ y $b = (2,3,4,5,6,7,8,9,10,11)$, obtendremos como vector $\frac{c'}{c} = (7,15,2,1,6,12,8,3)$ si tomamos $\beta = 1$. En cambio, tomando $\frac{1}{\beta} = 2$ si se puede realizar ya que $\beta = \frac{1}{2} = 9$ no se corresponde con ninguna coordenada de $\frac{c'}{c}$.

Ejemplo 6.4. Consideramos un Reed-Solomon generalizado sobre $\mathbb{F}_{13}$, $GRS_6(a,b)$, para el cual tomamos los vectores $a = (0,1,2,3,4,5,6,7,8,9)$ y $b = (2,3,4,5,6,7,8,9,10,11)$ que suponemos desconocidos.

Vamos a realizar la construcción de un criptosistema de McEliece mediante el empleo del código $GRS_6(a,b)$. Para ello, tomamos una matriz de permutación cuadrada aleatoria de tamaño 10, P, y una matriz invertible cuadrada aleatoria de tamaño 6, S:

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$S = \begin{pmatrix} 5 & 7 & 5 & 3 & 4 & 8 \\ 1 & 12 & 1 & 1 & 5 & 6 \\ 12 & 2 & 10 & 10 & 11 & 1 \\ 0 & 9 & 7 & 9 & 5 & 11 \\ 7 & 7 & 6 & 7 & 7 & 9 \\ 0 & 0 & 0 & 12 & 2 & 4 \end{pmatrix}$$

A partir de estas matrices, generamos la matriz pública del criptosistema de McEliece, que denotamos G_p:

$$G_p = \begin{pmatrix} 4 & 6 & 12 & 10 & 6 & 7 & 10 & 3 & 5 & 11 \\ 0 & 1 & 6 & 2 & 6 & 12 & 8 & 1 & 0 & 1 \\ 10 & 9 & 12 & 11 & 2 & 7 & 7 & 0 & 8 & 11 \\ 9 & 11 & 2 & 0 & 7 & 12 & 12 & 2 & 6 & 3 \\ 11 & 5 & 7 & 1 & 7 & 7 & 6 & 9 & 12 & 2 \\ 3 & 1 & 4 & 0 & 7 & 7 & 10 & 6 & 2 & 10 \end{pmatrix}$$

Ahora realizamos el cifrado de un mensaje cualquiera, $m = (8, 0, 5, 5, 2, 3)$, para ello introducimos sobre el codificado 2 errores, correspondientes a la máxima capacidad correctora del código $GRS_6(a, b)$, obteniendo $y = (4, 5, 10, 7, 11, 10, 9, 5, 10, 10)$.

Vamos a recuperar dicho mensaje tras realizar el ataque de filtración contra la estructura del código, es decir, después de recuperar el código $GRS_6(a, b)$ de partida.

En primer lugar tenemos que observar que para este caso, la dimensión es mayor que la mitad de la longitud y por tanto tenemos que recurrir al código dual $GRS_6(a, b)^{\perp} = GRS_4(a, b')$. Para ello, empleamos como matriz generatriz, G_{dual}, la matriz de control correspondiente al código con matriz generatriz G_p, la cual expresamos en forma estándar para facilitar el resto de cuentas:

$$G_{dual} = \begin{pmatrix} 1 & 0 & 0 & 0 & 9 & 8 & 7 & 6 & 10 & 9 \\ 0 & 1 & 0 & 0 & 11 & 2 & 1 & 4 & 10 & 1 \\ 0 & 0 & 1 & 0 & 4 & 3 & 12 & 11 & 11 & 9 \\ 0 & 0 & 0 & 1 & 3 & 5 & 12 & 8 & 2 & 11 \end{pmatrix}$$

Ahora calculamos la matriz generatriz del código recortado en la primera posición, G_{s10}, en la segunda posición, G_{s01}, y en ambas posiciones, G_{s11}, las cuales tienen los siguientes valores:

$$G_{s10} = \begin{pmatrix} 1 & 0 & 0 & 11 & 2 & 1 & 4 & 10 & 1 \\ 0 & 1 & 0 & 4 & 3 & 12 & 11 & 11 & 9 \\ 0 & 0 & 1 & 3 & 5 & 12 & 8 & 2 & 11 \end{pmatrix}$$

$$G_{s01} = \begin{pmatrix} 1 & 0 & 0 & 9 & 8 & 7 & 6 & 10 & 9 \\ 0 & 1 & 0 & 4 & 3 & 12 & 11 & 11 & 9 \\ 0 & 0 & 1 & 3 & 5 & 12 & 8 & 2 & 11 \end{pmatrix}$$

$$G_{s11} = \begin{pmatrix} 1 & 0 & 4 & 3 & 12 & 11 & 11 & 9 \\ 0 & 1 & 3 & 5 & 12 & 8 & 2 & 11 \end{pmatrix}$$

También calculamos las matrices generatrices que denotamos G_{p10} y $G_{p10_{s01}}$, del código punteado en la primera posición a partir del código $GRS_6(a, b)^{\perp}$ y del código punteado en la primera posición a partir del código que se construye empleando la matriz G_{s01} como generatriz:

$$G_{p10} = \begin{pmatrix} 1 & 0 & 0 & 0 & 11 & 4 & 1 & 5 & 3 \\ 0 & 1 & 0 & 0 & 11 & 6 & 4 & 8 & 5 \\ 0 & 0 & 1 & 0 & 11 & 1 & 6 & 3 & 8 \\ 0 & 0 & 0 & 1 & 11 & 8 & 5 & 4 & 1 \end{pmatrix}$$

$$G_{p10_{s01}} = \begin{pmatrix} 1 & 0 & 0 & 11 & 6 & 4 & 8 & 5 \\ 0 & 1 & 0 & 11 & 1 & 6 & 3 & 8 \\ 0 & 0 & 1 & 11 & 8 & 5 & 4 & 1 \end{pmatrix}$$

Ahora a partir de los códigos con matrices generatrices G_{p10} y G_{s10}, realizando un proceso semejante al del ejemplo 6.3, el cual no mostramos, obtenemos el código con dimensión 1 cuya matriz generatriz nos proporciona los valores de las coordenadas del vector c: $c = (1, 7, 12, 10, 5, 8, 8, 7, 1)$.

De igual forma, a partir de los códigos con matrices generatrices $G_{p10_{s01}}$ y G_{s11}, realizando también el proceso anterior obtenemos el vector $c' = (1, 7, 12, 12, 5, 2, 12, 8)$.

Después de la obtención de los vectores c y c', realizamos la división por coordenadas de c' entre c_1, donde c_1 se corresponde con el vector c al que eliminamos la primera coordenada. De esta forma obtenemos $\frac{c'}{c_1} = (2, 6, 9, 5, 12, 10, 11, 8)$ y como $1 \notin \frac{c'}{c}$ podemos tomar el valor $\beta_0 = 1$ y realizar la evaluación de cada una de sus coordenadas sobre $y \longmapsto = \frac{1}{1-y}$ para recuperar todas las coordenadas del vector a excepto las dos primeras, lo cual no supone ningún problema al estar estas posiciones fijados sus valores.

Por tanto obtenemos el vector $a = (0, 1, 12, 5, 8, 3, 7, 10, 9, 11)$ y ahora realizando la división de c entre a^3 recuperamos todas las coordenadas del vector b' excepto b'_1, es decir, $b' = (b'_1, 1, 6, 8, 2, 5, 12, 5, 7, 8)$. Para recuperar la primera coordenada, vamos dando valores a b'_1 hasta obtener un vector b' que se encuentre en el código $GRS_4(a, b')$ de partida.

Para finalizar, nos falta calcular el vector b que nos permita recuperar el código $GRS_6(a, b)$ de partida. Para ello, simplemente construimos el código $GRS_9(a, b')$, y su matriz de control se corresponde con el vector b que buscamos.

Entonces, calculando ahora el código $GRS_6(a, b)$ y aplicando su algoritmo de corrección de errores sobre el vector y, obtenemos $c = (2, 5, 10, 7, 11, 4, 9, 5, 10, 10)$ y realizando la descodificación a partir de una matriz cuadrada invertible construida mediante columnas de G_p recuperamos el mensaje de partida, $m = (8, 0, 5, 5, 2, 3)$.

Capítulo 7

Reducción de claves propuesta por Gaborit

7.1. Introducción

Sabemos que el criptosistema McEliece es resistente frente a ataques realizados mediante ordenadores cuánticos, por lo que podría ser un criptosistema que sustituya en un futuro a el RSA, logaritmo discreto o cualquier otro sistema de seguridad criptográfica actual. Sin embargo, el gran tamaño de la clave pública del criptosistema de McEliece supone un gran problema, debido a que la memoria necesaria para guardar las claves se incrementa notablemente según vamos empleando criptosistemas mas extensos.

Por ello, en esta parte vamos a ver un método para reducir la clave pública del sistema de McEliece, el cual fue descrito por Gaborit en el artículo expuesto en 2004 [10] y aunque dicho método fue criptoanalizado de forma exitosa, tiene importancia al ser el primero en intentar reducir la clave empleando la estructura de los códigos cíclicos y cuasiciclicos. Estos y otros códigos están siendo empleados actualmente con ciertas mejoras para crear un criptosistema estándar de McEliece en un concurso organizado por el NIST (National Institute of Standars and Technology) llamado "Post-Quantum Cryptography Standardization Process"[30]. En este documento, presentamos en el capítulo 9 una versión innovadora del criptosistema de McEliece que trata de reducir el tamaño de las claves empleando la idea principal que describimos en este capítulo pero usando códigos de producto de matrices que veremos en el capítulo 8.

Antes de empezar, comenzamos introduciendo los códigos cíclicos y cuasiciclicos junto con sus propiedades mas elementales y necesarias para la explicación del método realizado por Gaborit. Si se desea conocer mas acerca de este tipo de códigos se recomiendan las referencias [12] y [13].

Definición 7.1. Sea C un código lineal, diremos que C es un código cíclico si y solo si para cualquier palabra del código $c = (c_1, ..., c_n) \in C$ se tiene que $c' \in C$ donde $c' = (c_n, c_1, ..., c_{n-1})$.

La siguiente proposición puede considerarse como una definición alternativa de un código cíclico, donde se representan las palabras del código como polinomios en lugar de vectores:

Proposición 7.2. *Dado* $\mathbb{F}_q[x]/\langle x^n - 1\rangle$, *consideramos el siguiente isomorfismo*

$$\varphi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q[x]/\langle x^n - 1\rangle$$

$$(c_1, ..., c_n) \longmapsto c_1 + \cdots + c_n x^{n-1}$$

Se tiene que $C \subset \mathbb{F}_q[x]/\langle x^n - 1\rangle \cong \mathbb{F}_q^n$ *es un código cíclico si y solo si* C *es un ideal de* $\mathbb{F}_q[x]/\langle x^n - 1\rangle$.

Demostración.
$\Rightarrow$) Sea C un código cíclico, la aplicación φ nos permite identificar C como un subconjunto de $\mathbb{F}_q[x]/\langle x^n - 1\rangle$. Ademas, C es un subespacio lineal cerrado para las operaciones de suma y producto por escalares. Si vemos que si $c \in C$ y $x \in \mathbb{F}_q[x]/\langle x^n - 1\rangle$ entonces $cx \in C$ tenemos demostrado que C es un ideal, ya que tenemos entonces que si $f(x) = a_1 + a_2 x + ... + a_n x^{n-1} \in \mathbb{F}_q[x]/\langle x^n - 1\rangle$ es un polinomio cualquiera de grado menor que n, $a_i \in \mathbb{F}_q \; \forall i = 1, ..., n$, se tiene que $cf \in C$ porque al tener que $cx \in C$ tenemos que $cx^2 \in C$ y de forma sucesiva tenemos que $cx^i \in C \; \forall i = 0, ..., n-1$. Ademas, también tenemos que $a_{i+1} c x^i \in C$ por ser C cerrado para el producto por escalares y entonces $cf(x) = a_1 c + a_2 cx + ... + a_{n+1} cx^n \in C$ al ser C cerrado para la suma de palabras del código C. Empleando el isomorfismo φ tenemos que

$$cx = c_1 x + ...c_n x^n = c_1 x + ...c_n (x^n - 1) + c_n = c_n + ... + c_{n-1} x^{n-1} \in C$$

que es lo que faltaba por demostrar para tener que C es un ideal de $\mathbb{F}_q[x]/\langle x^n - 1\rangle$.

$\Leftarrow$) Sea C un ideal de $\mathbb{F}_q[x]/\langle x^n - 1\rangle$, es evidente que es un subespacio lineal, debido a la definición de ideal. Ademas, sea $c = (c_1, ..., c_n) \in C$, sabemos que $cx \in C$ por ser C un ideal y razonando igual que en la otra implicación, tenemos que $cx = (c_n, c_1, ..., c_{n-1})$ y por tanto C es un código cíclico como queríamos. □

Ahora definimos los códigos cuasi-cíclicos, que son mas generales que los códigos cíclicos.

Definición 7.3. Sea C un código lineal de parámetros $[n, k]$, diremos que C es un código cuasi-cíclico de orden s, donde s es un divisor de n, si todo cambio cíclico de s coordenadas es de nuevo una palabra del código, es decir, si $(c_1, ..., c_n) \in C$ entonces se tiene que $(c_{n-s+1}, ..., c_n, c_1, ..., c_{n-s}) \in C$.

Nota 7.4. Si el orden del código cuasi-cíclico es $s = 1$ entonces se corresponde con un código cíclico.

Nota 7.5. Si C es un código cíclico de longitud n, entonces C también es cuasi-cíclico de orden s, para cualquier s que divida n.

Lema 7.6. *Si C es un código cuasi-cíclico de orden s, entonces $C^{\perp}$ también es un código cuasi-cíclico de orden s.*

Proposición 7.7. *Sea C un código cuasi-cíclico de parámetros $[n, k]$ de orden s con $n = rs$. Entonces existe una matriz G de tamaño $k' \times n$, donde $k' \geq k$, que genera el código C y es de la forma*

$$G = \begin{pmatrix} A_1 & A_2 & \cdots & A_r \\ A_r & A_1 & \cdots & A_{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_2 & A_3 & \cdots & A_1 \end{pmatrix}$$

donde A_i es una matriz de tamaño $\frac{k'}{r} \times s \ \forall i = 1, ..., r$.

Demostración. Sea $c = (c_1, ..., c_n) \in C$ una palabra del código C arbitraria. Como C es cuasi-cíclico de orden s, separamos c en $r = \frac{n}{s}$ partes iguales obteniendo $(c_1, ..., c_s), (c_{s+1}, ..., c_{2s}), ..., (c_{n-s+1}, ..., c_n)$. Llamamos $A_i^1 = (c_{s(i-1)+1}, ..., c_{is})$ con $1 \leq i \leq r$ y consideramos la matriz G^1 definida de la forma

$$G^1 = \begin{pmatrix} A_1^1 & A_2^1 & \cdots & A_r^1 \\ A_r^1 & A_1^1 & \cdots & A_{r-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ A_2^1 & A_3^1 & \cdots & A_1^1 \end{pmatrix}$$

la cual, genera un código, que llamamos C_1. Si $C_1 = C$ entonces estaría probado y sabemos cuales son los A_i.

Si $C_1 \neq C$ consideramos otra palabra del código $c' \in C$ cualquiera que cumpla que $c' \notin C_1$. Separando c' en r partes iguales de igual forma que hemos realizado para c y tomando las matrices A_i^2 formadas respectivamente por la matrices A_i^1 (en este primer caso son vectores) a las que se añade una fila formada por cada una de las partes en que hemos dividido el vector c', es decir

$$A_i^2 = \begin{pmatrix} A_i^1 \\ c'^i \end{pmatrix}$$

donde $c'^i = (c'_{s(i-1)+1}, ..., c'_{is})$ con $1 \leq i \leq r$ hace referencia a la parte i-esima en que hemos dividido el vector c'.

A partir de estas matrices, A_i^2 podemos construir la matriz G^2 que genera un código C_2. Realizando la misma comprobación que antes, continuamos realizando este proceso iterativo. Ademas, como el rango de las matrices G^j va incrementándose de forma estricta según incrementa j, esto nos garantiza que podemos parar el proceso en la iteración j_0, la cual verifica que $k' = rj_0 \geq k$ ya que la matriz G^{j_0} genera el código C. □

Nota 7.8. Obviamente, para conocer la matriz G descrita en la proposición 7.7, la cual nos permite conocer el código C, es suficiente con conocer las matrices $A_1, ..., A_n$.

Nota 7.9. La matriz G genera el código C pero no tiene porque cumplirse que sea una matriz generatriz del código C, ya que esto solo ocurrirá en el caso particular en que $k' = k$.

Proposición 7.10. *Sea C un código cuasi-cíclico de parámetros $[n, k]$ y orden s, con $n = rs$, entonces existen subcódigos cuasi-cíclicos contenidos estrictamente en C, de orden s con dimensión mayor o igual a $k - r = k - \frac{n}{s}$*

Demostración. Sea $C^\perp$ el código dual de C, que por el lema 7.6 sabemos que es también un código cuasi-cíclico y de orden s. Tomamos un vector x en $\mathbb{F}_q^n \setminus C^\perp$ y construimos la matriz G_x formada por la matriz generatriz de $C^\perp$ a la que añadimos r filas, que son el vector x junto con las $r - 1$ posibles permutaciones de orden s que se realizan al vector x. Llamamos C_x al código generado por la matriz G_x el cual es por construcción un código cuasi-cíclico de orden s y de dimension a lo sumo $n - k + r$. Tomando ahora el código dual de C_x tenemos entonces que es también un código cuasi-cíclico de orden s y con dimensión mayor o igual a $k - r$ como queríamos ver. □

Proposición 7.11. *Sea C un código cuasi-cíclico de parámetros $[n, k]$ entonces al menos 2^{k-r} códigos distintos pueden ser construidos por la proposición 7.10, donde $r = \frac{n}{s}$.*

Demostración. Según la demostración realizada en la proposición 7.10, es evidente que la cantidad de los distintos subcodigos es equivalente a la cantidad de los distintos códigos duales de estos subcodigos.

Como la unión de todos los posibles duales debe ser todo el espacio y cada código tiene dimension a lo sumo de $n - k + r$, entonces hay al menos $2^{n-(n-k+r)} = 2^{k-r}$ subcodigos cuasi-cíclicos distintos

□

7.2. Método de reducción de claves

A partir de las propiedades descritas, es evidente que si partimos de un código cuasi-cíclico C de parámetros $[n, k]$ y orden s, podemos obtener una matriz generadora,

G, de la forma vista en la proposición 7.7, la cual se puede obtener con las matrices A_i descritas, y por tanto el tamaño de la clave pública se reduce de forma notable.

Si consideramos ademas una permutación de s coordenadas, la cual es semejante a considerar una matriz de permutación de tamaño s que denotamos por π. Aplicamos dicha permutación sobre cada fila de las $\frac{k'}{r}$ posibles de cada una de las matrices A_i, que tienen s coordenadas cada una, $i = 1, ..., r$. Obtenemos entonces una matriz enmascarada que llamamos G', la cual se sigue pudiendo obtener únicamente del conocimiento de las matrices A_i y la matriz de permutación π:

$$G' = \begin{pmatrix} A_1\pi & A_2\pi & \cdots & A_r\pi \\ A_r\pi & A_1\pi & \cdots & A_{r-1}\pi \\ \vdots & \vdots & \ddots & \vdots \\ A_2\pi & A_3\pi & \cdots & A_1\pi \end{pmatrix}$$

Por tanto, nuestra clave pública será el conjunto de matrices A_i sobre las que se ha aplicado la permutación π, a partir de las cuales podemos obtener la matriz G'.

Obviamente, esta matriz no siempre nos permite realizar el cifrado de los mensajes, ya que $k' \geq k$. Por tanto, tenemos que obtener una matriz generatriz (enmascarada) y única del código cuasi-cíclico que estamos empleando, que denotaremos M, para poder realizar el cifrado.

Para su obtención, consideramos el vector v_1 formado por las coordenadas de la primera fila de la matriz G', es decir, por las coordenadas de la primera fila de todas las matrices $A_i\pi$. Por tanto, el vector v_1 tiene longitud $n = rs$. Ahora consideramos la matriz M_1 (para este primer caso es un vector) formada por el vector v_1. Entonces tomamos el vector v_1' obtenido al realizar una permutación de s coordenadas aplicada a las r divisiones del vector v_1 de s coordenadas cada una. Si el vector v_1' no pertenece al código generado por M_1 entonces se considera la matriz M_2 obtenida por la adicción de la fila v_1' a la matriz, M_1 que teníamos anteriormente. Repitiendo este proceso de permutación de s coordenadas sobre el vector v_1 r veces obtenemos la matriz M_r obtenida a partir del vector v_1. Repetimos el mismo proceso ahora para el vector v_2 formado por las coordenadas de la segunda fila de G', teniendo en cuenta en adicción que para cada vector v_2' obtenido al realizar una permutación, debe cumplir que es independiente de las filas que tenemos en la matriz M_r actualmente. Reiterando este proceso para las $\frac{k'}{r}$ primeras filas de la matriz G', que se corresponden con todas las filas de las matrices A_i, obtenemos como resultado la matriz M, que es única y cumple que tiene un total de k filas, n columnas y genera el código cuasi-cíclico enmascarado. Esta matriz es la que emplearemos para cifrar los mensajes.

En conclusión, tenemos que el método de criptosistema de McEliece descrito pre-

senta las siguientes propiedades:

7.2.1. Generación de claves

La clave publica consiste en las primeras $\frac{k'}{r}$ filas de la matriz G' correspondientes con las matrices $A_i\pi$ con $i = 1, ..., r$, es decir, el conjunto de matrices, A_i, que generan la matriz G sobre las que se ha aplicado la matriz de permutación, π. Ademas, la clave pública también consta del numero de errores que es capaz de corregir el código, t, así como del orden del código cuasi-cíclico que estamos empleando.

La clave privada consiste simplemente en la matriz de permutación π que hemos empleado para enmascarar las matrices A_i con $i = 1, ..., r$.

7.2.2. Cifrado

El proceso de cifrado de un mensaje x se realiza calculando la matriz M a partir de la matriz G' que se obtiene de la clave pública. Una vez que poseemos la matriz M, el cifrado consiste simplemente en calcular $c = xM + e$, donde e es un vector de errores de peso máximo t, siendo t la capacidad correctora del código que estamos empleando. Entonces el vector c es el mensaje cifrado, y es por tanto el vector que enviamos al receptor.

7.2.3. Descifrado

El proceso de descifrado de un mensaje c se realiza también calculando la matriz M a partir de la matriz G' que se obtiene de la clave pública. Entonces mediante la clave privada, calculamos la matriz de permutación inversa total, Π^{-1}, siendo Π la matriz que realiza la misma permutación cada s columnas un total de r veces:

$$\Pi = \begin{pmatrix} \pi & 0 & \cdots & 0 \\ 0 & \pi & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \pi \end{pmatrix}$$

Entonces obtenemos $c\Pi^{-1} = xM\Pi^{-1} + e\Pi^{-1}$. Ahora, aplicamos el algoritmo de corrección de hasta t errores del código que estamos empleando obteniendo $xM\Pi^{-1}$, y entonces recuperamos el mensaje original, x.

7.2.4. Tipo de códigos sugeridos

Una vez descrito el criptosistema, tenemos que encontrar una familia densa de códigos que sean cuasi-cíclicos de orden s, resistentes a los ataques conocidos en la actualidad como por ejemplo, el ataque de filtración que hemos explicado en el capítulo 6, el ataque de Sidelnikov y Shestakov [8] o el ataque de Stern [14] entre otros existentes. También deben ser códigos que se puedan codificar y descodificar rápidamente. Para ello Gaborit propuso en [10] usar la familia de códigos BCH, los cuales verifican las condiciones impuestas, y en especial, el uso de códigos BCH primitivos de longitud $n = 2^m - 1$ ya que obtenemos códigos con mejores parámetros que los códigos BCH que no son primitivos.

Por tanto, se utilizaban el conjunto de subcódigos cuasi-cíclicos de orden s de un código BCH, los cuales se construían como se mostró en la proposición 7.10

7.3. Criptoanálisis del método

El desarrollo del criptosistema de McEliece aplicando el método visto en la sección 7.2 habría supuesto un gran avance para el empleo del McEliece, debido a que solventa su principal problema, que es el elevado tamaño de las claves.

Sin embargo, en 2010 Otmani, Tillich y Dallot presentan un artículo[15] en el que criptoanalizan exitosamente el criptosistema que utiliza dicho método y emplea los subcodigos de un código BCH primitivo, ademas de otro criptoanalisis que emplea códigos cuasi-cíclicos con matrices de control de baja densidad (LDPC).

Nosotros nos centraremos en mostrar el criptoanalisis realizado sobre los subcódigos de un código BCH primitivo, cuya debilidad se encuentra en la obtención de una gran cantidad de ecuaciones lineales que tienen que cumplir las entradas de la matriz de permutación secreta debido a que se emplea como código secreto subcódigos de un código BCH, el cual puede considerarse conocido aunque en principio no lo sea como veremos mas adelante.

A partir de esto, es posible recuperar la matriz secreta de permutación, Π, que se esta empleando en dicho criptosistema y por tanto obtenemos la clave privada del criptosistema.

Veamos el método que hay que emplear para obtener la matriz de permutación Π.

Sea C_0 un código BCH primitivo cuasi-cíclico de orden s, de longitud $n = rs$ y dimension $k = rk_0$, el cual admite una matriz de control, H_0, de tamaño $(n-k) \times n$ la cual puede considerarse que es conocida. Esto es debido a que hay muy pocos códigos BCH primitivos para un conjunto de parámetros establecidos, $[n, m, t]$ a priori, en

el criptosistema. Esto sucede ya que la cantidad de estos códigos esta acotada superiormente por el numero de polinomios primitivos de grado m, de los cuales no hay muchos. Por tanto, se puede realizar el método que vamos a mostrar con todos los códigos BCH primitivos posibles para esos parámetros.

Tomamos ahora C, un subcodigo del código C_0. Podemos considerar que la dimensión de C es $k - r = r(k_0 - 1)$ debido a la proposición 7.10 en la que se muestra como construir los subcodigos a partir de un código cuasi-cíclico dado . Ademas, se verifica que

$$\forall c \in C,\ H_0 \times c^T = 0 \tag{7.1}$$

ya que si $c \in C$ entonces también se verifica que $c \in C_0$

Una vez visto esto, si consideramos una matriz de permutación, π, que permuta las s columnas de cada matriz A_i, $i = 1, ..., r$, que proporcionan la matriz generadora del código C, denotada por G segun lo visto en la proposición 7.7, suponemos sin perdida de generalidad que $k' = k$ (si esto no ocurriera habría que considerar en lugar de la matriz G, la matriz M generada por el proceso descrito en la sección 7.2 a partir de la matriz G) y denotamos ahora por Π a la matriz de permutación mostrada en la sección 7.2.3.

Entonces, para alguna matriz de permutación, π, de las $s!$ posibles se verifica que la matriz $G^\pi = G \cdot \Pi$ se corresponde con la matriz generatriz publica, si el subcodigo C escogido se corresponde con la clave privada que se esta empleando.

Obviamente, es inviable comprobar todas las permutaciones para todos los subcodigos posibles, ya que sabemos por la proposición 7.10 que para cada código BCH de los posibles existen al menos 2^{k-r} subcodigos. Por tanto, para cada código BCH binario tendríamos una complejidad de al menos $s! \cdot 2^{k-r}$ operaciones a los que faltaría aun añadir las operaciones necesarias para comprobar que la permutación elegida se corresponde con la clave privada.

Sin embargo, gracias a la ecuación 7.1 tenemos que se debe verificar que

$$H_0 \cdot (G^\pi \cdot \Pi^{-1})^T = 0$$

lo cual nos proporciona un sistema lineal, donde únicamente desconocemos la matriz Π^{-1}, que nos permite recuperar la permutación π que enmascaraba el código. Para ello, cada fila de la matriz pública G^π nos proporciona $(n - k)$ ecuaciones lineales binarias que debe cumplir la matriz Π^{-1}. Por tanto, como tenemos un total de $k - r$ filas en la matriz G^π, obtenemos un total de $(k - r)(n - k)$ ecuaciones lineales para s^2 incógnitas, siendo $s = \frac{n}{r}$.

Como tenemos que $k = rk_0$, siendo k_0 la dimension de las matrices A_i, $\forall i = 1, ..., r$ y $n = rs$, entonces tenemos s^2 incógnitas para $r^2(k_0 - 1)(s - k_0)$ ecuaciones y como

para los dos conjuntos de parámetros establecidos por Gaborit para los niveles de seguridad mostrados en el articulo [10] se cumple que $r > s$, entonces aunque pueda haber muchas ecuaciones linealmente dependientes, como tenemos que el número de ecuaciones es superior al numero de incógnitas, el espacio vectorial que obtenemos como solución es prácticamente siempre de dimension 1, es decir, obtendremos una única solución, la cual se corresponde con la permutación secreta, π, que enmascaraba el subcodigo C.

Después de realizar con éxito el criptoanalisis del método desarrollado por Gaborit en [15], Gaborit junto con Otmani, Cayrel y Berger desarrollaron en el artículo [16] un criptosistema basado principalmente en el desarrollado en el artículo [10], sobre el cual se realizan una serie de cambios que impedían que el criptosistema se pueda criptoanalizar exitosamente por el ataque que hemos explicado.

Sin embargo, en 2011 surgen también otros ataques frente a dicho criptosistema que se muestran en los artículos [17], [18], [19] y [20].

Capítulo 8

Códigos de productos de matrices

8.1. Introducción

Los códigos de producto de matrices fueron introducidos por Blackmore y Norton en un artículo desarrollado en 2001 [21]. Dichos códigos surgieron a partir de la generalización de la construcción de Plotkin, también conocida como $(u \mid u+v)$.

La principal característica de estos códigos es que se construyen a partir de un conjunto de códigos mas pequeños, lo cual resulta interesante para la teoría de codificación.

Definición 8.1. Sea $A = [a_{ij}]$ una matriz $M \times N$ con entradas en $\mathbb{F}_q$, con $M \leq N$ y sean $C_1, ..., C_M$ códigos de longitud n sobre el cuerpo $\mathbb{F}_q$. Diremos que el código de producto de matrices, denotado por $[C_1 \cdots C_M] \cdot A$ es el conjunto de todos los productos de matrices $[c_1 \cdots c_M] \cdot A$, donde $c_i \in C_i$ hace referencia a un vector columna de tamaño n, $\forall i = 1, ..., M$. Por tanto, las palabras del código $[C_1 \cdots C_M] \cdot A$ son matrices $n \times N$ de la forma

$$c = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1M} \\ c_{21} & c_{22} & \cdots & c_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nM} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix} =$$

$$\begin{pmatrix} c_{11}a_{11} + \cdots + c_{1M}a_{M1} & \cdots & c_{11}a_{1N} + \cdots + c_{1M}a_{MN} \\ \vdots & \ddots & \vdots \\ c_{n1}a_{11} + \cdots + c_{nM}a_{M1} & \cdots & c_{n1}a_{1N} + \cdots + c_{nM}a_{MN} \end{pmatrix}$$

Nota 8.2. Podemos ver que la j-esima columna en la palabra del código también se puede escribir para simplificar como $\sum_{i=1}^{M} c_{hi}a_{ij}$ para $1 \leq h \leq n$, donde $1 \leq j \leq N$.

Ademas, si denotamos por $c_i = [c_{1i}, ..., c_{ni}]^T$ entonces podemos escribir la j-esima columna como $\sum_{i=1}^{M} c_i a_{ij}$

Entonces, las palabras del código de $[C_1 \cdots C_M] \cdot A$ se pueden ver como vectores escritos de la forma

$$c = [\sum_{i=1}^{M} c_i a_{i1}, \sum_{i=1}^{M} c_i a_{i2}, ..., \sum_{i=1}^{M} c_i a_{iN}] \in \mathbb{F}_q^{nN} \tag{8.1}$$

donde la matriz se ha escrito en el conocido como orden de tamaño de mayor columna. En esta forma, observamos que el código tiene tamaño nN

Entonces si $C_1, ..., C_M$ son códigos lineales, se puede encontrar una matriz generatriz, G, para el código de producto de matrices, cuya definición es evidente a partir de como hemos realizado la construcción de los códigos de producto de matrices:

$$G = \begin{pmatrix} G_1 a_{11} & \cdots & G_1 a_{1N} \\ \vdots & \ddots & \vdots \\ G_M a_{M1} & \cdots & G_M a_{MN} \end{pmatrix}$$

donde G_i es la matriz generatriz del código C_i para $i = 1, ..., M$.

Ademas, por la matriz generatriz del código podemos ver que la dimensión del código de producto de matrices es $k = k_1 + ... + k_M$ donde k_i es la dimensión del código C_i, $i = 1, ..., M$.

Aquí hay que remarcar que esta igualdad solamente ocurre en el caso en que la matriz A tenga el máximo rango, lo cual supondremos a partir de ahora salvo que se diga lo contrario. En el caso en que la matriz A no tenga máximo rango entonces se tiene que $k < k_1 + ... + k_M$.

Ejemplo 8.1. Consideramos tres códigos lineales C_1, C_2 y C_3 todos ellos sobre $\mathbb{F}_2$ y de longitud $n = 3$ constituidos por los siguientes elementos:

$$C_1 = \{(0,0,0), (1,1,1)\}$$

$$C_2 = \{(0,0,0), (1,0,0)\}$$

$$C_3 = \{(0,0,0), (1,0,1), (1,1,0), (0,1,1)\}$$

Por tanto, las dimensiones son respectivamente $k_1 = 1$, $k_2 = 1$ y $k_3 = 2$.

Consideramos también la matriz A definida de la forma

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

El código de producto de matrices, que denotaremos $C = [C_1\, C_2\, C_3] \cdot A$, cuyo cardinal es, según la proposición 8.6 que se muestra mas adelante, $|[C_1\, C_2\, C_3] \cdot A| = |C_1||C_2||C_3| = 2 * 2 * 4 = 16$, donde cada palabra del código tiene una longitud de $3 * 3 = 9$. Calculamos ahora dichas palabras del código de producto de matrices, donde realizaremos el cálculo mediante el producto de matrices en 3 de ellas

$$c_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$c_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$c_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

En lugar de expresarlas como matrices, las expresaremos normalmente como vectores, en la forma vista en (8,1), es decir

$$c_1 = (0,0,0,\ \ 0,0,0,\ \ 0,0,0)$$
$$c_2 = (0,0,0,\ \ 0,0,0,\ \ 1,0,1)$$
$$c_3 = (0,0,0,\ \ 0,0,0,\ \ 1,1,0)$$

Para obtener todas las palabras del código emplearemos el código desarrollado en *Sage* que se muestra en [1, capítulo 7] del trabajo fin de grado de informática. A partir de este, obtenemos que las palabras del código C son

$$c_1 = (0,0,0,\ \ 0,0,0,\ \ 0,0,0)$$
$$c_2 = (0,0,0,\ \ 0,0,0,\ \ 1,0,1)$$
$$c_3 = (0,0,0,\ \ 0,0,0,\ \ 1,1,0)$$
$$c_4 = (1,1,1,\ \ 1,1,1,\ \ 1,1,1)$$

$$
\begin{aligned}
c_5 &= (0,0,0,\ \ 1,0,0,\ \ 1,0,0)\\
c_6 &= (1,1,1,\ \ 0,1,1,\ \ 0,1,1)\\
c_7 &= (1,1,1,\ \ 1,1,1,\ \ 0,1,0)\\
c_8 &= (0,0,0,\ \ 1,0,0,\ \ 0,0,1)\\
c_9 &= (1,1,1,\ \ 0,1,1,\ \ 1,1,0)\\
c_{10} &= (0,0,0,\ \ 0,0,0,\ \ 0,1,1)\\
c_{11} &= (1,1,1,\ \ 1,1,1,\ \ 1,0,0)\\
c_{12} &= (0,0,0,\ \ 1,0,0,\ \ 1,1,1)\\
c_{13} &= (1,1,1,\ \ 0,1,1,\ \ 0,0,0)\\
c_{14} &= (1,1,1,\ \ 1,1,1,\ \ 0,0,1)\\
c_{15} &= (0,0,0,\ \ 1,0,0,\ \ 0,1,0)\\
c_{16} &= (1,1,1,\ \ 0,1,1,\ \ 1,0,1)
\end{aligned}
$$

Ahora vamos a realizar el cálculo de una matriz generatriz de C, G. Para ello, tomamos una matriz generatriz de cada uno de los códigos C_i, $i = 1,2,3$

$$
G_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad G_2 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \quad G_3 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}
$$

Entonces, a partir de las matrices G_1, G_2 y G_3, la matriz generatriz de C es

$$
G = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
$$

Definición 8.3. Sea A una matriz $M \times N$, denotamos $A(j_1, ..., j_t)$ a la matriz cuadrada de tamaño t formada a partir de las t primeras filas de la matriz A y las columnas $j_1, ..., j_t$, donde $1 \leq j_1 < \cdots < j_t \leq N$ con $1 \leq t \leq M$.

Nota 8.4. Si $A(1, ..., M)$ es no singular, la matriz $N \times M$ cuyas M primeras filas son las de la matriz $A(1, ..., M)^{-1}$ y las $N - M$ ultimas filas son nulas, es una matriz inversa de la matriz A.

Proposición 8.5. *Sean $C_1, ..., C_M$ códigos lineales y A una matriz $M \times N$.*

1) *Si A_π es una matriz obtenida al realizar una permutación de las filas de la matriz A, entonces se tiene que $[C_1 \cdots C_M] \cdot A$ es un código equivalente a $[C_{\pi(1)} \cdots C_{\pi(M)}] \cdot A_\pi$.*

2) *Si A_ρ es una matriz obtenida al realizar una permutación de las columnas de la matriz A, entonces se tiene que $[C_1 \cdots C_M] \cdot A$ es un código equivalente a $[C_1 \cdots C_M] \cdot A_\rho$*

Demostración.

1) Si π es la permutación realizada, como las palabras del código de $C_1, ..., C_M$ se corresponden con columnas para el código de producto de matrices, las cuales se multiplican por la matriz A_π que ha sufrido las mismas permutaciones para las filas de la matriz A. Por tanto se realizan las mismas multiplicaciones y sumas tanto en $[C_1 \cdots C_M] \cdot A$ como en $[C_{\pi(1)} \cdots C_{\pi(M)}] \cdot A_\pi$, obteniendo palabras del código equivalentes al tener únicamente cambiado el orden de las coordenadas y por tanto son códigos equivalentes.

2) Si ρ es una permutación sobre las columnas de A, tenemos que únicamente se cambia el orden de las coordenadas de las palabras del código de producto de matrices y por tanto, obtenemos que $[C_1 \cdots C_M] \cdot A$ y $[C_1 \cdots C_M] \cdot A_\rho$ son también códigos equivalentes.

□

Proposición 8.6. *Sea A una matriz $M \times N$. Si tenemos una matriz formada por M columnas de A que es no singular entonces se verifica que*

$$|[C_1 \cdots C_M] \cdot A| = |C_1| \cdots |C_M|$$

Demostración. Podemos suponer sin perdida de generalidad que $A(1, ..., M)$ es no singular, por el resultado obtenido en la proposición 8.5. Consideramos la aplicación

$$\phi : C_1 \times \cdots \times C_M \to [C_1 \cdots C_M] \cdot A$$
$$(c_1, ..., c_M) \longmapsto [c_1 \cdots c_M] \cdot A$$

Si vemos que ϕ es una biyección estaría probado.

Claramente ϕ es sobreyectiva según la construcción realizada de la aplicación y la definición del código de producto de matrices. En cuanto a la inyectividad, consideramos que $[c_1 \cdots c_M] \cdot A = [c'_1 \cdots c'_M] \cdot A$.

Como $A(1, ..., M)$ es no singular, existe $A(1, ..., M)^{-1}$ y también existe la matriz A^{-1}, de tamaño $N \times M$.

Por tanto, A es no singular y entonces multiplicando a ambos lados por A^{-1} obtenemos que $[c_1 \cdots c_M] = [c'_1 \cdots c'_M]$. Por ello, tenemos que $(c'_1, ..., c'_M) = (c_1, ..., c_M)$ como queríamos. □

A partir de lo que hemos visto hasta ahora, podemos observar que hemos obtenido tanto la longitud del código $[C_1 \cdots C_M] \cdot A$, como su cardinalidad y dimensión. Sin embargo, para obtener la distancia mínima de dicho código vamos a introducir primero el concepto de matriz no singular por columnas y matriz triangular, que nos será útil mas adelante.

Definición 8.7. Dada una matriz A de tamaño $M \times N$, diremos que A es una matriz no singular por columnas (NSC) si $A(j_1, ..., j_t)$ es no singular para cada $1 \leq t \leq M$ y $1 \leq j_1 < \cdots < j_t \leq N$.

Obviamente se verifica que toda matriz no singular por columnas es también una matriz no singular, sin embargo el reciproco no es cierto en muchas ocasiones, por ejemplo, si consideramos la matriz

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

es evidente que es no singular ya que $det(A) = -1$, y no es NSC ya que es necesario que en la primera fila todos sus elementos sean no nulos.

Otro ejemplo de matriz no singular que no es NSC sería la matriz A empleada en el ejemplo 8.1.

Definición 8.8. Diremos que un código de producto de matrices es NSC si la matriz A asociada al código es NSC.

Para enunciar la siguiente proposición es necesario indicar la construcción de los códigos C_{R_i}, con $i = 1, ..., M$. Fijado un i, el código C_{R_i} es el que tiene como matriz generatriz, que denotaremos por A_i, la formada por las i primeras filas de la matriz A.

Proposición 8.9. *Dada una matriz A de tamaño $M \times N$, verifica que es NSC si y solo si los códigos C_{R_i} son MDS $\forall i = 1, ..., M$.*

Demostración. Si la matriz A es NSC sabemos por definición que $A(j_1, ..., j_t)$ es no singular para cada $1 \leq t \leq M$ y $1 \leq j_1 < \cdots < j_t \leq N$.

Por otro lado, fijado $r \in \{1, ..., M\}$, consideramos la matriz generatriz A_r, correspondiente al código C_{R_r}. Como la matriz A es NSC, es evidente que todos los determinantes correspondientes a las matrices de los menores principales de A_r son distintos de cero, luego la matriz A_r es no singular. Esto nos permite concluir que el código C_{R_r} es MDS, ya que al ser A_r no singular es de máximo rango y por tanto sus columnas son linealmente independientes. Sabemos por [22, Corolario 3, capítulo 11] que las

columnas de A_r son linealmente independientes si y solo si el código C_{R_r} es MDS. Como esta demostración es válida para $r \in \{1, ..., M\}$ se tiene entonces lo que se quería demostrar. □

Definición 8.10. Diremos que una matriz A es triangular si existe alguna permutación por columnas, π, que transforma la matriz en una matriz triangular superior, es decir, si $a_{i\pi(j)} = 0 \, \forall i > \pi(j)$

Proposición 8.11. *Una matriz triangular NSC A de tamaño $M \times N$ tiene exactamente $i-1$ ceros en la i-esima fila, con $1 \leq i \leq M$.*

Demostración. Obviamente, al ser una matriz triangular, en la fila i-esima debe tener al menos $i-1$ ceros.

Veamos ahora que no puede tener mas que $i-1$ ceros. Supongamos que existe una fila, i, que tiene mas de $i-1$ ceros. Si consideramos la matriz $A(j_1, ..., j_i)$, donde las posiciones $j_1, ..., j_i$ son las posiciones que hacen referencia a un termino nulo dentro de la fila i, es decir, $a_{ij_1} = ... = a_{ij_i} = 0$. Tenemos que dicha matriz tiene una fila nula y por tanto $det(A(j_1, ..., j_i)) = 0$, lo cual es absurdo ya que por hipótesis la matriz A es NSC.

□

Ahora vamos a introducir la proposición que nos permitirá conocer la distancia mínima de un código de producto de matrices, pero antes de ello, denotamos $R_i = (a_{i1}, ..., a_{iN}) \in \mathbb{F}_q^N$, es decir, R_i es un vector de tamaño N formado por los elementos de la i-esima fila de la matriz A, los cuales pertenecen a $\mathbb{F}_q$.

A partir de ello, construimos ahora los códigos C_{R_i} generados a partir de los vectores R_j con $j = 1, ..., i$, es decir, generados por las i primeras filas de la matriz A como hemos definido anteriormente. Por tanto, $C_{R_i} = \langle R_1, ..., R_i \rangle$, y denotamos por D_i la distancia del código C_{R_i}.

Proposición 8.12. *Dado un código de producto de matrices $C = [C_1 \cdots C_M] \cdot A$, entonces se cumple la siguiente desigualdad para la mínima distancia:*

$$d(C) \geq min\{d_1 * D_1, d_2 * D_2, ..., d_M * D_M\}$$

donde d_i hace referencia a la distancia del código C_i y D_i a la distancia del código C_{R_i}.

Demostración. Tomamos c una palabra cualquiera del codigo C, entonces podemos escribir $c = [c_1 \ \cdots c_M] \cdot A$ donde c_i hace referencia a un vector columna de tamaño n.

Consideramos que $c_i \neq 0 \, \forall i = 1, ..., r$ y $c_i = 0 \, \forall i = r+1, ..., M$ entonces tenemos que

$$c = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1r} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & c_{2r} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nr} & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{pmatrix} =$$

$$\begin{pmatrix} c_{11}a_{11} + \cdots + c_{1r}a_{r1} & \cdots & c_{11}a_{1N} + \cdots + c_{1r}a_{rN} \\ \vdots & \ddots & \vdots \\ c_{n1}a_{11} + \cdots + c_{nr}a_{r1} & \cdots & c_{n1}a_{1N} + \cdots + c_{nr}a_{rN} \end{pmatrix}$$

Entonces podemos observar que los elementos de c están formados por componentes de la matriz A desde la primera fila hasta la fila R_r. Por tanto si denotamos $(c_{h1}, ..., c_{hM})$ la fila h de $[c_1 \cdots c_M]$, tenemos que $(c_{h1}, ..., c_{hM}) \cdot A \in C_{R_r}\ \forall h = 1, ..., n$. Como $c_i \neq 0\ \forall i = 1, ..., r$ entonces debe tener al menos d_r coordenadas distintas de cero.

A partir de esto consideramos $c_{i_u r} \neq 0$ siendo $u = 1, ..., d_r$ entonces tenemos que $(c_{i_u 1}, ..., c_{i_u M}) \cdot A$ pertenece a C_{R_r} y es distinto de cero, ya que la única opción de que fuera 0 sería si la matriz A no tiene rango máximo, entonces el peso mínimo debe ser mayor o igual que D_r y como C_r tiene distancia mínima d_r entonces se tiene que $d(c) \geq d_r * D_r$. Por tanto, variando el r, obtenemos que $d(C) \geq min\{d_1 * D_1, d_2 * D_2, ..., d_M * D_M\}$ como queríamos. □

Nota 8.13. Como C_{R_i} esta generado a partir de los vectores $R_1, ..., R_i$, es decir, a partir de las i primeras filas de la matriz A, es evidente que $C_{R_i} \subset C_{R_{i+1}}$ y por tanto también que $D_i \geq D_{i+1}$. Por tanto, para obtener un código de producto de matrices con buenos parámetros es conveniente elegir los códigos C_j de manera que la distancia mínima de estos se vaya incrementando o al menos sea igual, es decir, $d_{j+1} \geq d_j$, para obtener de esta forma la mayor distancia posible en el código de producto de matrices a partir de los códigos que lo generan, lo cual sabemos que permite corregir una mayor cantidad de errores. También es aconsejable coger C_{j+1} con una dimensión menor o igual que la de C_j.

Por tanto, para que pueda ocurrir lo descrito anteriormente será necesario que C_1 tenga una dimensión grande y que el último de los códigos tenga una distancia mínima grande.

Además de intentar maximizar la distancia mínima del código de producto de matrices, como esta viene delimitada a partir de una desigualdad, es importante también saber exactamente como de grande es la distancia mínima del código, para conocer de esta forma, la cantidad exacta de errores máxima que podemos corregir.

Por ello, vamos a ver a continuación dos formas de obtener la igualdad en la desigualdad que tenemos de la distancia.

Para la primera forma, consideramos que los códigos $C_1, ..., C_M$ son encajados, es decir, $C_M \subset C_{M-1} \subset \cdots \subset C_1$. Estos códigos verifican que $d_M \geq d_{M-1} \geq \cdots \geq d_1$, luego no estamos restringiendo de forma excesiva el conjunto de códigos a escoger al ser del tipo deseado para maximizar la distancia del código de producto de matrices. Para los conjuntos de códigos con esta propiedad se verifica el siguiente teorema, con la igualdad deseada para la distancia mínima del código de producto de matrices.

Teorema 8.14. *Dado un código de producto de matrices $C = [C_1 \ \cdots \ C_M] \cdot A$ donde C_i con $i = 1, ..., M$ son códigos que verifican $C_M \subset C_{M-1} \subset \cdots \subset C_1$ y A es una matriz de tamaño $M \times N$. Entonces se cumple que*

$$d(C) = min\{d_1 * D_1, d_2 * D_2, ..., d_M * D_M\}$$

Demostración. Por la proposición 8.12 sabemos que se cumple que

$$d(C) \geq min\{d_1 * D_1, d_2 * D_2, ..., d_M * D_M\}$$

Por tanto, si vemos que existe una palabra del código de producto de matrices verificando la igualdad, obtendremos la igualdad para la distancia del código de producto de matrices.

Por ello, tomamos $c \in C$, la cual, es de la forma $[c_1 \ \cdots \ c_M] \cdot A$ cumpliendo, para un $r \in \{1, ..., M\}$ fijo, que $c_1 = c_2 = ... = c_r$, con $\omega(c_1) = d_r$ y $c_{r+1} = ... = c_M = 0$. La hipótesis de $c_1 = c_2 = ... = c_r$ es valida al ser los códigos encajados.

Si tomamos ahora una palabra, b, del código C_{R_r} de peso D_r, podemos expresar b como combinación lineal de los vectores R_i con $i = 1, ..., r$, es decir, $b = \sum_{i=1}^{r} w_i R_i$, siendo $w_i \in \mathbb{F}_q$. Tomando $c_i' = w_i c_i$ tenemos que

$$(\sum_{i=1}^{M} a_{i1} c_i', ..., \sum_{i=1}^{M} a_{iN} c_i') = (\sum_{i=1}^{M} c_i a_{i1} w_i, ..., \sum_{i=1}^{M} c_i a_{iN} w_i)$$

Ahora teniendo en cuenta que $c_1 = c_2 = ... = c_r$, $c_{r+1} = ... = c_M = 0$ podremos sacar factor común a c_1 y el sumatorio esta limitado superiormente por r ya que los siguientes términos serían nulos. Además como $R_i = (a_{i1}, ..., a_{iN})$ tenemos que

$$(\sum_{i=1}^{M} c_i a_{i1} w_i, ..., \sum_{i=1}^{M} c_i a_{iN} w_i) = c_1 (\sum_{i=1}^{r} a_{i1} w_i, ..., \sum_{i=1}^{r} a_{iN} w_i) = c_1 (\sum_{i=1}^{r} w_i R_i) = c_1 b$$

que es una palabra del código C con peso $d_r D_r$

□

En cuanto a la segunda forma, imponemos que la matriz A sea NSC y triangular, lo cual nos permite obtener la igualdad en la desigualdad de la distancia para los códigos de producto de matrices que tenemos como podemos ver en el siguiente teorema.

Teorema 8.15. *Si A es una matriz $M \times N$ NSC y el código NSC es $C = [C_1 \cdots C_M] \cdot A$, entonces se cumple que:*

1) $|C| = |C_1| \cdots |C_M|$

2) $d(C) \geq d^* = min\{Nd_1, (N-1)d_2, ..., (N-M+1)d_M\}$

3) Si ademas A es también una matriz triangular entonces $d(C) = d^*$

Demostración.

1) Es evidente a partir de la proposición 8.6 ya que sabemos que si A es una matriz NSC entonces A es una matriz no singular.

2) Como A es una matriz NSC, sabemos por la proposición 8.9 que los códigos C_{R_i} son MDS $\forall i = 1, ..., M$ y por tanto sabemos por la cota de Singleton que la distancia es $D_i = N - i + 1$ y entonces por la proposición 8.12 tenemos lo que queríamos.

3) Sea A una matriz triangular. Entonces como sabemos por la proposición 8.5 que permutaciones de las columnas de la matriz A proporcionan códigos de producto de matrices equivalentes, podemos suponer, sin perdida de generalidad que A es una triangular superior.

   Sea $(N-t+1)d_t$ el valor mas pequeño de los valores $(N-i+1)d_i$ y sean $c, c' \in C$ con $c_t, c'_t \in C_t$ tal que $d(c_t, c'_t) = d_t$ y $c_i = c'_i \; \forall i \neq t$. Entonces $d(c, c') \leq d(c_t a_{tt}, c'_t a_{tt}) + d(c_t a_{tt+1}, c'_t a_{tt+1}) + \cdots + d(c_t a_{tN}, c'_t a_{tN}) \leq (N-t+1)d_t$.

   Luego hemos encontrado dos palabras del código C que tienen como distancia mínima a lo sumo d^*. Entonces por la parte 2) del teorema tenemos la igualdad.

□

Ejemplo 8.2. Consideramos los códigos C_1, C_2, C_3 y la matriz A del ejemplo 8.1.

Claramente, vemos que la matriz A es triangular, sin necesidad de realizar ninguna permutación. Sin embargo, no es NSC al existir un determinante de orden 2 que es nulo. Por tanto, la desigualdad obtenida en la proposición 8.12 no es igualdad según lo visto en el teorema 8.15.

Las distancias de los códigos considerados se obtienen en este caso fácilmente a partir del mínimo peso de Hamming de las palabras de cada código, las cuales son respectivamente $d_1 = 3$, $d_2 = 1$, $d_3 = 2$.

Por tanto, si reordenamos los códigos de la forma $C_1' = C_2$, $C_2' = C_3$ y $C_3' = C_1$ obtenemos que $d_1' \leq d_2' \leq d_3'$ y la distancia del código de producto de matrices $C = [C_1'\, C_2'\, C_3'] \cdot A$ es $d(C) \geq d^* = \min\{3*1, 2*2, 1*3\} = 3$, donde d^* es el máximo valor que se puede alcanzar a partir de los códigos C_1, C_2, C_3 dados, lo cual se puede verificar rápidamente en el ejemplo, ya que si tomamos otra ordenación de los códigos donde C_1 no sea el último código escogido, tendremos que la última multiplicación dentro del mínimo a realizar para obtener la distancia vale o 2 o 1 y por tanto la distancia obtenida es menor. Se puede razonar de forma semejante para el código intermedio, lo cual nos lleva a la ordenación elegida al comienzo para tener la máxima distancia posible del código de producto de matrices generado a partir de los códigos C_1, C_2, C_3 considerados.

De hecho podemos observar, que no existe ninguna matriz cuadrada de tamaño 3 formada por elementos de $\mathbb{F}_2$ que sea NSC ya que, según la proposición 8.11, la primera fila de la matriz no puede tener ningún elemento nulo y la segunda fila solo puede tener un cero, por tanto, siempre existirá un determinante de orden dos formado únicamente por elementos con valor 1, es decir, su determinante será nulo.

8.2. Descodificación

Después de mostrar los códigos de productos de matrices así como la obtención de sus parámetros, de forma exacta, si empleamos una matriz A no singular por columnas y triangular gracias al teorema 8.15, o si utilizamos códigos encajados para la generación de los códigos de producto de matrices según lo visto en el teorema 8.14.

Ahora nos preguntamos si podemos emplear los códigos de productos de matrices en el criptosistema de McEliece. Hasta el momento, hemos visto como obtener una matriz generatriz para estos códigos, así como realizar una codificación. Por tanto, lo que nos falta por obtener es como realizar una descodificación para dichos códigos.

Por ello, vamos a mostrar un algoritmo que permita realizar la descodificación de los códigos de producto de matrices, para el cual es necesario que los códigos que empleamos para la obtención de los códigos de producto de matrices sean encajados, así como que la matriz A sea no singular por columnas, lo cual, vimos anteriormente que no es una condición tan restrictiva ya que queremos formar códigos de producto de matrices con la mayor mínima distancia posible. También es necesario que los códigos empleados para generar el código de producto de matrices dispongan cada uno de

ellos de un algoritmo de descodificación.

Teniendo en cuenta lo anterior, consideramos, $C = [C_1 \cdots C_M] \cdot A$, un código de producto de matrices formado por la matriz A no singular por columnas y por los códigos $C_1, ..., C_M$ encajados y de longitud n, donde cada código C_i, con $i = 1, ..., M$, posee un algoritmo de descodificación, que denotamos por DC_i, el cual es capaz de corregir hasta un máximo de $t_i = \frac{d_i - 1}{2}$ errores de una palabra del código C_i

Si tomamos ahora una palabra del código producto de matrices $c \in C$, se cumple que $c = (\sum_{j=1}^{M} a_{j1}c_j, ..., \sum_{j=1}^{M} a_{jN}c_j)$ donde $c_j \in C_j \; \forall j = 1, ..., M$.

Como los elementos de la matriz A pertenecen a $\mathbb{F}_q$ tenemos que $\sum_{j=1}^{M} a_{jh}c_j$, con $h = 1, ..., N$, es una combinación lineal de una palabra de cada código C_i, con $i = 1, ..., M$, y como $C_M \subset \cdots \subset C_2 \subset C_1$, todas son palabras del código C_1, por tanto, $\sum_{j=1}^{M} a_{jh}c_j$, con $h = 1, ..., N$, también es una palabra del código C_1.

Debido a esto, es evidente que podríamos emplear el algoritmo de descodificación DC_1 sobre los N bloques que constituyen la palabra de código c. Pero como tenemos que $C_1 \supset \cdots \supset C_M$ se verifica que $d_1 \leq d_2 \leq \cdots \leq d_M$ y entonces también se cumple que $t_1 \leq t_2 \leq \cdots \leq t_M$. Por tanto, seriamos capaces de corregir como máximo t_1 errores en cada bloque de una palabra del código, es decir, $t_1 N$ errores en total. Sin embargo, estos errores deben están distribuidos de forma que haya a lo sumo t_1 errores en cada bloque, si esto no ocurre, la descodificación fallaría. Por tanto, de esta forma para poder garantizar que la descodificación siempre se realiza exitosamente, el número máximo de errores que tenemos que tomar es de t_1 y por tanto, la capacidad correctora del código producto de matrices es de t_1 errores.

Por ello, vamos a mostrar un método de descodificación que nos permita corregir siempre la máxima capacidad correctora del código, es decir, $t = \frac{d(C) - 1}{2}$, siendo $d(C)$ la distancia del código de producto de matrices C, obtenida a partir del teorema 8.14 o el teorema 8.15.

Para este método vamos a emplear los algoritmos de descodificación de los M códigos que generan el código de producto de matrices. Mostramos a continuación el algoritmo que permite descodificar una palabra recibida $y = c + e$ donde $c \in C$ y e es un vector de errores de longitud nN con un peso menor o igual a la capacidad máxima correctora del código, es decir, $\omega(e) \leq t$. Para ello, dicho algoritmo debe recibir como parámetros tanto la matriz A como los M códigos junto con sus algoritmos de descodificación, ademas de la palabra y que queremos descodificar.

Algoritmo 8.16.

$1 : y' = y;$

2 : $A' = A$;
3 : $for\ \{i_1, ..., i_M\} \subset \{1, ..., N\}$:
4 : $y = y'$;
5 : $A = A'$;
6 : $for\ j = 1, ..., M$:
7 : $y_{i_j} = DC_j(y_{i_j})$;
8 : $if\ y_{i_j} =$ "fallo" :
9 : *break* #Rompemos el for y tomamos otro $\{i_1, ..., i_M\}$ en la linea 2
10 : $for\ k = j + 1, ..., M$:
11 : $y_{i_k} = y_{i_k} - \frac{a_{ji_k}}{a_{ji_j}} y_{i_j}$;
12 : $columna_{i_k}(A) = columna_{i_k}(A) - \frac{a_{ji_k}}{a_{ji_j}} columna_{i_j}(A)$;
13 : Recuperamos $(c_1, ..., c_M)$;
14 : $y = [c_1, ..., c_M] \cdot A$;
15 : $if\ y \in C$ and $\omega(y - y') \leq [\frac{d(C)-1}{2}]$:
16 : return y;

Si y es la palabra recibida, el algoritmo considera $\{i_1, ..., i_M\} \subset \{1, ..., N\}$ un subconjunto ordenado de indices, de forma que el vector de errores $e = (e_1, ..., e_N)$ satisfaga que $\omega(e_{i_j}) \leq t_j\ \forall j \in \{1, ..., M\}$. Si el subconjunto ordenado de indices no verifica esto en todos los indices, por ejemplo, supongamos que no se cumple para el indice i_{j_0}, entonces el correspondiente algoritmo de descodificación DC_{j_0} no es capaz de corregir todos los errores que tiene ese bloque, y por tanto, puede ocurrir que no encuentre ninguna palabra del código C_{j_0} que se encuentre a distancia menor que t_{j_0} del bloque $y_{i_{j_0}}$ y en ese caso, podemos suponer que devuelve una respuesta de "fallo", o que por el contrario si encuentre una palabra que pertenezca al código C_{j_0} y entonces el algoritmo nos devolverá como resultado una palabra del código que no es la correcta. En esta situación, si no existe otro indice para el que el respectivo algoritmo de descodificación proporcione como respuesta "fallo", podemos percatarnos de que la descodificación realizada es errónea comprobando si la palabra descodificada, constituida por todos los bloques descodificados, es una palabra del código de producto de matrices C y comprobando también que no se han corregido mas de t errores. En cualquiera de los dos casos, habría que considerar otro subconjunto distinto de indices y realizar el mismo procedimiento.

Ademas, podemos observar que es suficiente con tomar un subconjunto de indices de M elementos dentro de los N posibles ya que como la matriz A es no singular por columnas, de las N columnas, M son linealmente independientes y las $N - M$ restantes se pueden obtener como combinación lineal de las otras.

Si denotamos para cada $i = 1, ..., N$ por $y_i = \sum_{j=1}^{M} a_{ji}c_j + e_i$ el i-esimo bloque de la palabra recibida y. Entonces, como el bloque i_1 tiene un vector de error menor que la capacidad correctora del código C_1 y ademas al ser códigos incrustados sabemos que cualquier bloque de c es una palabra del código C_1. Empleando el algoritmo de descodificación DC_1 podemos obtener el bloque i_1 sin errores, ademas del bloque i_1 del vector de errores.

Para eliminar los errores en el resto de bloques vamos tomando el vector $y^{k)}$, con $k = 2, ..., M$, formado por las componentes

$$y_i^{k)} = y_i^{k-1)} - \frac{a_{(k-1)i}^{k-1)}}{a_{(k-1)i_{k-1}}^{k-1)}}(y_{i_{k-1}}^{k-1)} - e_{i_{k-1}}) = \sum_{j=k}^{M} a_{ji}^{k)}c_j + e_i \ \forall i \neq i_1, ..., i_{k-1}$$

donde

$$a_{ji}^{k)} = a_{ji}^{k-1)} - \frac{a_{(k-1)i}^{k-1)}}{a_{(k-1)i_{k-1}}^{k-1)}}a_{ji_{k-1}}^{k-1)}$$

y

$$y_{i_1}^{k)} = y_{i_1}^{k-1)}, ..., y_{i_{k-2}}^{k)} = y_{i_{k-2}}^{k-1)}, y_{i_{k-1}}^{k)} = y_{i_{k-1}}^{k-1)} - e_{i_{k-1}}$$

Como por construcción, si $i \in \{1, ..., M\} \setminus \{i_1, ..., i_{k-1}\}$, tenemos que el bloque i-esimo de $y^{k)}$ es una palabra del código C_k mas el i-esimo bloque del vector error, usando el algoritmo de descodificación DC_k sobre el i_k-esimo bloque, obtenemos el bloque de error i_k y $\sum_{j=k}^{M} a_{ji_k}^{k)}c_j$

Ahora, como tenemos los bloques de error e_i, con $i \in \{i_1, .., i_M\}$, podemos recuperar los bloques de la palabra del código $\sum_{j=k}^{M} a_{ji}^{k)}c_j$, con $i \in \{i_1, .., i_M\}$, y por tanto, tenemos el vector $(\sum_{j=k}^{M} a_{ji_1}^{k)}c_j, ..., \sum_{j=k}^{M} a_{ji_M}^{k)}c_j)$ sin errores, el cual es igual a el resultado que se obtiene al multiplicar $[c_1 \cdots c_M] \cdot A(i_1, ..., i_M)$, siendo $A(i_1, ..., i_M)$ la submatriz de tamaño M de la matriz A formada a partir de las columnas $i_1, ..., i_M$ de la matriz A. Como la matriz $A(i_1, ..., i_M)$ es invertible al tener rango máximo por ser A no singular por columnas, podemos obtener los vectores $c_1, ..., c_M$.

Por último, realizando la multiplicación $[c_1 \cdots c_M] \cdot A$ podemos recuperar los $N - M$ bloques restantes del vector c, es decir, los $N - M$ bloques restantes del vector y sin errores, como queríamos.

Sin embargo, para que el algoritmo sea correcto es necesario comprobar que los vectores $y^{k)}$ estén bien definidos, es decir, que $a_{ki_k}^{k)} \neq 0$ para cada $k = 2, ..., M$, para evitar dividir entre cero.

Si definimos la matriz $A^{k)} = (a_{ij}^{k)})$, de tamaño $M \times N$, para $k = 2, ..., M$. Considerando que $A^{1)} = A$, estas matrices se van construyendo de forma recursiva a partir de la matriz anterior de la siguiente forma:

$$a_i^{k)} = a_i^{k-1)} - \frac{a_{(k-1)i}^{k-1)}}{a_{(k-1)i_{k-1}}^{k-1)}} a_{i_{k-1}}^{k-1)} \ \forall i \notin \{i_1, ..., i_{k-1}\}$$

donde el vector $a_i^{k)}$ hace referencia a la columna i-esima de la matriz $A^{k)}$

Podemos observar que, si fijamos $k = 2, ..., M$, estas operaciones introducen $N - (k-1)$ elementos nulos en la fila $k-1$ de $A^{k)}$ y por tanto el menor de $A^{k)}$ formado por las primeras k filas y las columnas $i_1, ..., i_k$ se corresponde con la matriz

$$\begin{pmatrix} a_{1i_1}^{k)} & 0 & 0 & \cdots & 0 \\ a_{2i_1}^{k)} & a_{2i_2}^{k)} & 0 & \cdots & 0 \\ a_{3i_1}^{k)} & a_{3i_2}^{k)} & a_{3i_3}^{k)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{ki_1}^{k)} & a_{ki_2}^{k)} & a_{ki_3}^{k)} & \cdots & a_{ki_k}^{k)} \end{pmatrix}$$

la cual es una matriz triangular inferior si los indices elegidos son crecientes, es decir, si tenemos que $i_h < i_{h+1} \ \forall h = 1, ..., M-1$. Si los indices no son crecientes, obtenemos una matriz triangular inferior tras realizar una permutación de las columnas de la matriz.

Además, podemos observar que el determinante de dicha matriz es $a_{1i_1}^{k)}, ..., a_{ki_k}^{k)}$, el cual sabemos que es no nulo por ser A una matriz no singular por columnas, y por tanto $a_{ki_k}^{k)} \neq 0$

Por tanto, hemos visto que la descodificación y corrección de errores realizada por el algoritmo siempre es exitosa si encontramos un subconjunto ordenado de indices que satisfaga que $\omega(e_{i_j}) \leq t_j \ \forall j \in \{1, ..., M\}$.

Entonces, para que el algoritmo funcione siempre, nos falta comprobar que para cualquier vector de error, e, que verifique que su peso sea menor que la capacidad correctora del código de producto de matrices, es decir, que $\omega(e) \leq t$, siempre podemos encontrar un subconjunto ordenado de indices cumpliendo esa restricción.

Teorema 8.17. *Sea $C = [C_1 \cdots C_M] \cdot A$ un código de producto de matrices con A una matriz no singular por columnas y los códigos $C_i \subset \mathbb{F}_q^n \ \forall i = 1, ..., M$ encajados. Si $e = (e_1, ..., e_N) \in F_q^{Nn}$ es un vector de error con peso $\omega(e) \leq t = \lfloor \frac{d(C)-1}{2} \rfloor$, entonces existe un subconjunto*

ordenado de indices $\{i_1, ..., i_M\} \subset \{1, ..., N\}$ *tal que* $\omega(e_{i_j}) \leq t_j = \lfloor \frac{d_j - 1}{2} \rfloor \; \forall j = 1, ..., N$, *siendo* d_j *la distancia mínima del código* C_j.

Demostración. Para realizar la demostración, veremos primero que se verifica para un único indice y después realizaremos inducción para comprobar que el resultado se verifica hasta considerar un subconjunto de indices ordenado cuyo cardinal sea de M elementos, lo cual concluiría la demostración.

Sea $j_0 \in \{1, ..., M\}$ fijo, veamos que existe un i_{j_0} tal que $\omega(e_{i_{j_0}}) \leq t_{j_0}$. Para ello, razonamos por reducción al absurdo, supongamos que no existe ningún $i_{j_0} \in \{1, ..., N\}$ con $\omega(e_{i_{j_0}}) \leq t_{j_0}$, entonces se cumple que $\omega(e_i) > t_{j_0} \; \forall i = 1, ..., N$ y por tanto tenemos que

$$\omega(e) \geq \frac{N d_{j_0}}{2} \geq \frac{(N - (j_0 - 1)) d_{j_0}}{2} > \frac{(N - (j_0 - 1)) d_{j_0} - 1}{2} \geq$$
$$\geq \left\lfloor \frac{(N - (j_0 - 1)) d_{j_0} - 1}{2} \right\rfloor \geq \left\lfloor \frac{d(C) - 1}{2} \right\rfloor$$

lo cual es absurdo ya que por hipótesis tenemos que $\omega(e) \leq t = [\frac{d(C) - 1}{2}]$.

Ahora supongamos que la propiedad se verifica para el subconjunto de indices $\{i_1, ..., i_{j-1}\} \subset \{1, ..., N\}$ y tamaño $j - 1 < M$ y veamos que se cumple también para el subconjunto de indices de tamaño j. Supongamos por reducción al absurdo que no existe ningún i_h con $\omega(e_{i_h}) \leq t_h = [\frac{d_h - 1}{2}] \; \forall h = j, ..., N$, entonces se verifica que $\omega(e_i) > t_j \; \forall i \in \{1, ..., N\} \setminus \{i_1, ..., i_{j-1}\}$ y por tanto tenemos que

$$\omega(e) \geq \sum_{k=1}^{j-1} \omega(e_{i_k}) + \frac{(N - (j - 1)) d_j}{2} > \sum_{k=1}^{j-1} \omega(e_{i_k}) + \frac{(N - (j - 1)) d_j - 1}{2} \geq$$

$$\geq \left\lfloor \frac{(N - (j - 1)) d_j - 1}{2} \right\rfloor \geq \left\lfloor \frac{d(C) - 1}{2} \right\rfloor$$

por lo que llegamos a la misma contradicción ya que por hipótesis tenemos que $\omega(e) \leq t = \lfloor \frac{d(C) - 1}{2} \rfloor$.

□

Ejemplo 8.3. Como vimos en el ejemplo 8.2 que no es posible elegir una matriz cuadrada de tamaño 3 NSC formada por elementos de $\mathbb{F}_2$, vamos a elegir una matriz A de tamaño 3 sobre el cuerpo $\mathbb{F}_{13}$:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Podemos observar que dicha matriz es triangular y ademas es NSC. En cuanto a los códigos que emplearemos para realizar la construcción del código de producto de matrices, usaremos tres códigos Reed-Solomon, todos ellos obviamente sobre el cuerpo $\mathbb{F}_{13}$.

Ademas, como deben ser códigos encajados, el vector empleado para la construcción de los tres códigos es $a = \mathbb{F}_{13} \setminus \{0\}$, luego su longitud es 12 en los tres casos y la dimensión va decreciendo en 2, es decir, la dimensión del código C_1 es $k_1 = 6$, la de C_2 es $k_2 = 4$ y la de C_3 es $k_3 = 2$.

Como vimos en la proposición 3.3 que los códigos Reed-Solomon son MDS, tenemos que la distancia mínima de dichos códigos es $d_1 = 7$, $d_2 = 9$ y $d_3 = 11$. Por tanto, su capacidad correctora es $t_1 = 3$, $t_2 = 4$ y $t_3 = 5$.

En cuanto al algoritmo de descodificación, podemos observar que el elemento 2 es un elemento primitivo del cuerpo $\mathbb{F}_{13} \setminus \{0\}$. Entonces podríamos emplear el algoritmo de descodificación para códigos Reed-Solomon cíclicos visto en la sección 3.3. Sin embargo, emplearemos el método general visto en la misma sección, al estar dicho algoritmo ya implementado en *Sagemath* como se puede ver en [1, sección 3.3] del trabajo fin de grado de informática.

La matriz generatriz del código C_1 es:

$$G_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 4 & 9 & 3 & 12 & 10 & 10 & 12 & 3 & 9 & 4 & 1 \\ 1 & 8 & 1 & 12 & 8 & 8 & 5 & 5 & 1 & 12 & 5 & 12 \\ 1 & 3 & 3 & 9 & 1 & 9 & 9 & 1 & 9 & 3 & 3 & 1 \\ 1 & 6 & 9 & 10 & 5 & 2 & 11 & 8 & 3 & 4 & 7 & 12 \end{pmatrix}$$

y las matrices generatrices de C_2 y C_3, denotadas por G_2 y G_3, se corresponden con las matrices formadas únicamente por las 4 y 2 primeras filas de la matriz G_1 respectivamente.

Por tanto, el código de producto de matrices $C = [C_1\ C_2\ C_3] \cdot A$ tiene longitud $n = 36$, dimensión $k = 12$ y distancia mínima $d(C) = 11$.

Consideramos ahora el mensaje $m = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$ de longitud 12, el cual codificamos a partir de la matriz generatriz del código C, G, que obtenemos según lo visto en la nota 8.2 o realizando en su lugar, la codificación de las 6 primeras coordenadas del mensaje m con el código C_1, obteniendo el vector c_1, las 4 coordenadas siguientes con el código C_2, obteniendo el vector c_2 y las 2 ultimas con el código C_3, obteniendo el vector c_3 y ahora multiplicando la matriz formada por los vectores c_i como columnas, por la matriz A.

El mensaje codificado que obtenemos de cualquiera de estas dos formas es:

$c = (2, 11, 3, 2, 4, 5, 3, 0, 4, 5, 3, 10, 10, 12, 11, 9, 6, 2, 12, 3, 0, 10, 11, 6, 1, 11, 11, 1, 5, 8, 10, 2, 7, 4, 8, 7)$

Como la capacidad correctora de C es $t = 5$, vamos a introducir un vector de errores cuyo peso de Hamming sea 5:

$$e = (0, 3, 0, 2, 0, 0, 6, 0, 0, 7, 0, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

Por tanto, obtenemos que el vector con errores, $y = c + e$, es:

$y = (2, 1, 3, 4, 4, 5, 9, 0, 4, 12, 3, 8, 10, 12, 11, 9, 6, 2, 12, 3, 0, 10, 11, 6, 1, 11, 11, 1, 5, 8, 10, 2, 7, 4, 8, 7)$

Ahora vamos a aplicar el algoritmo 8.16 que vimos para realizar la descodificación de los códigos de productos de matrices. Para nuestro ejemplo podemos observar que al ser $M = N = 3$ siempre tomamos los 3 bloques, donde es posible realizar $3! = 6$ ordenaciones distintas para la línea 3 del algoritmo 8.16.

Elegimos en primer lugar la ordenación $\{i_1 = 1, i_2 = 2, i_3 = 3\}$. Para el primer valor del bucle for, $j = 1$, tenemos que realizar la descodificación del primer bloque $y_1 = (2, 1, 3, 4, 4, 5, 9, 0, 4, 12, 3, 8)$ empleando el algoritmo de descodificación del código C_1, DC_1.

Al emplear dicho algoritmo, obtenemos un mensaje de error, lo cual parece razonable al ser la capacidad correctora del código C_1 de $t_1 = 3$.

Debido a esto, descartamos la ordenación escogida. Obviamente, si escogiéramos la ordenación $\{i_1 = 1, i_2 = 3, i_3 = 2\}$, obtendríamos el mismo problema ya que se aplicaría para el primer bloque también DC_1.

Por tanto, escogemos ahora la ordenación $\{i_1 = 2, i_2 = 1, i_3 = 3\}$. Comenzamos a aplicar el algoritmo 8.16 desde la linea 6:

Para $j = 1$, tenemos que aplicar el algoritmo de descodificación DC_1 para el bloque $y_2 = (10, 12, 11, 9, 6, 2, 12, 3, 0, 10, 11, 6)$, para el cual obtenemos como resultado el mismo vector, es decir, $y_2 = y_2^{2)}$, ya que podemos observar que en el segundo bloque no hay ningún error (En un caso real obviamente no conocemos el error).

Dentro del siguiente bucle for, comenzamos con el valor $k = 2$ para el cual calculamos el bloque $y_1^{2)} = y_1 - \frac{1}{1} * y_2^{2)} = (5, 2, 5, 8, 11, 3, 10, 10, 4, 2, 5, 2)$. Calculamos la matriz A, para la que obtenemos

$$A^{2)} = \begin{pmatrix} 0 & 1 & 1 \\ 11 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

ya que en este caso, simplemente hay que restar la segunda columna a la primera.

Para $k = 3$ calculamos el bloque $y_3^{2)} = y_3 - \frac{1}{1} * y_2^{2)} = (4, 12, 0, 5, 12, 6, 11, 12, 7, 7, 10, 1)$. En cuanto a la matriz $A^{2)}$ actualizamos ahora su tercera columna obteniendo

$$A^{2)} = \begin{pmatrix} 0 & 1 & 0 \\ 11 & 2 & 12 \\ 0 & 0 & 1 \end{pmatrix}$$

Ahora para $j = 2$ tenemos que aplicar el algoritmo DC_2 para el bloque $y_1^{2)}$ pero obtenemos de nuevo otro error al aplicar el algoritmo.

Seleccionamos ahora la ordenación $\{i_1 = 2, i_2 = 3, i_3 = 1\}$. Para $j = 1$ tenemos que aplicar DC_1 para el bloque y_2 y obtenemos que vuelven a coincidir, es decir, $y_2 = y_2^{2)}$.

Dentro del siguiente bucle for, para $k = 2$ calculamos el bloque $y_3^{2)} = y_3 - \frac{1}{1} * y_2^{2)} = (4, 12, 0, 5, 12, 6, 11, 12, 7, 7, 10, 1)$. Calculamos la matriz A, para la que obtenemos

$$A^{2)} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 12 \\ 0 & 0 & 1 \end{pmatrix}$$

Para $k = 3$ calculamos el bloque $y_1^{2)} = y_1 - \frac{1}{1} * y_2^{2)} = (5, 2, 5, 8, 11, 3, 10, 10, 4, 2, 5, 2)$ y actualizamos la matriz $A^{2)}$ obteniendo

$$A^{2)} = \begin{pmatrix} 0 & 1 & 0 \\ 11 & 2 & 12 \\ 0 & 0 & 1 \end{pmatrix}$$

Ahora tenemos que $j = 2$, entonces hay que aplicar el algoritmo DC_2 para el bloque $y_3^{2)}$ y obtenemos $y_3^{3)} = (4, 12, 0, 5, 12, 6, 11, 12, 7, 7, 10, 1)$.

Tenemos ahora como único valor, $k = 3$, para el cual calculamos el bloque $y_1^{3)} = y_1^{2)} - \frac{11}{12} * y_3^{3)} = (10, 4, 5, 11, 0, 4, 1, 12, 3, 1, 11, 0)$ y tenemos como matriz $A^{3)}$

$$A^{3)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 12 \\ 11 & 0 & 1 \end{pmatrix}$$

Para terminar tomamos el valor $j = 3$ y entonces solo nos queda aplicar el algoritmo DC_3 sobre el bloque $y_1^{3)}$ obteniendo $y_1^{4)} = (10, 1, 5, 9, 0, 4, 8, 12, 3, 7, 11, 2)$.

Ahora a partir de los distintos bloques obtenidos podemos recuperar el vector de error e a partir de la unión de los bloques e_1, e_2 y e_3:

$$e_1 = y_1^{3)} - y_1^{4)} = (0,3,0,2,0,0,6,0,0,7,0,11)$$

$$e_2 = y_2 - y_2^{2)} = (0,0,0,0,0,0,0,0,0,0,0,0)$$

$$e_3 = y_3^{2)} - y_3^{3)} = (0,0,0,0,0,0,0,0,0,0,0,0)$$

A partir del error podemos recuperar ya el mensaje codificado sin errores, c, y después recuperar el mensaje m resolviendo, al igual que hemos hecho en los ejemplos de otros códigos lineales, un sistema.

Capítulo 9

McEliece con codigos de productos de matrices

9.1. Introducción

En el capítulo 8, hemos introducido los códigos de producto de matrices, para los cuales hemos indicado como obtener sus distintos parámetros.

Además, es posible construir el criptosistema de McEliece visto en la sección 5.1 empleando en lugar de los códigos de Goppa binarios, los códigos de producto de matrices gracias al algoritmo de descodificación de dichos códigos visto en la sección 8.2, los cuales deben cumplir unas restricciones para que se pueda emplear dicho algoritmo.

De todas formas, usaremos códigos de Goppa para la construcción del código de producto de matrices que emplearemos para generar el criptosistema de McEliece, ya que sabemos que los códigos de Goppa no han sido criptoanalizados exitosamente en la actualidad y por lo tanto son seguros.

Sin embargo, podemos observar que si empleamos los códigos de producto de matrices utilizando la matriz generatriz obtenida a partir de la forma vista en la nota 8.2, seguimos teniendo la principal desventaja del criptosistema de McEliece que tenemos independientemente del código que empleemos para su construcción, correspondiente al tamaño de las claves pública y privada. Sin embargo, vamos a presentar a continuación una nueva versión del criptosistema de McEliece, la cual permite reducir el tamaño de las claves pública y privada cuando empleamos para la construcción del criptosistema de McEliece un código de producto de matrices.

En el capítulo 7 y en [10] vimos como Gaborit realizó una reducción en las claves del criptosistema de McEliece mediante el empleo de códigos cuasi-cíclicos. Para

ello, generaba la matriz generatriz a partir de pequeñas submatrices. La versión que presentamos ahora trata de reducir las claves del criptosistema empleando una idea parecida a la empleada por Gaborit pero utilizando los códigos de productos de matrices, lo cual permite evitar el ataque mostrado en [15] como vemos a continuación en la nota 9.1.

Vimos que la matriz generatriz del código de producto de matrices se generaba a partir de las matrices generatrices de los subcodigos empleados y la matriz A. A partir de esto, vamos a ver como es posible generar la clave pública a partir de dicha matriz A y las matrices generatrices enmascaradas de los subcodigos.

Para ocultar dichas matrices generatrices, emplearemos para cada matriz generatriz, una matriz invertible aleatoria S' y una matriz de permutación π. Dicha matriz de permutación coincide para todas las matrices generatrices, y por tanto es una matriz de permutación similar a la empleada por el método de Gaborit vista en la sección 7.2.3, lo cual es posible al tener todos los subcodigos empleados en el código de producto de matrices la misma longitud. Sin embargo, las matrices invertibles S' son distintas incluso en el tamaño, ya que son matrices cuadradas cuyo tamaño debe coincidir con la dimensión del código cuya matriz generatriz queremos ocultar.

Nota 9.1. Aunque la propuesta presentada por Gaborit fue criptoanalizada exitosamente, permitiendo recuperar la matriz de permutación Π empleada para ocultar la matriz generatriz del código cuasi-cíclico, para los códigos de productos de matrices es posible evitar dicho ataque empleando subcodigos que no sean los códigos BCH, los cuales eran los que presentaban la debilidad que permitía realizar dicho ataque.

Para la presentación del nuevo criptosistema de McEliece empleamos para los subcodigos códigos de Goppa. Por tanto, es posible mostrar el criptosistema ocultando la matriz generatriz del código de producto de matrices únicamente con la matriz de permutación Π de igual forma que en la versión mostrada por Gaborit, pero sin poder realizar el ataque que lo hace vulnerable.

Sin embargo, nosotros emplearemos además de la matriz de permutación Π, una matriz invertible S construida a partir de todas las matrices invertibles empleadas para cada matriz generatriz de los subcodigos empleados, para preservar el esquema original del criptosistema de McEliece. Aunque obviamente, tomando como matriz invertible S la matriz identidad, obtenemos como caso particular la versión que solo emplea la matriz de permutación.

9.2. El nuevo criptosistema de McEliece

Vamos a introducir ahora como realizar la generación de claves, así como el proceso a seguir para realizar el cifrado y descifrado.

9.2.1. Generacion de claves

Para la generación de las claves, vamos a emplear un código producto de matrices $[C_1 \cdots C_M] \cdot A$ formado por una matriz A no singular por columnas cuadrada de tamaño M y M códigos de Goppa binarios encajados, todos ellos de longitud n, de los cuales conocemos un algoritmo de descodificación que hemos visto en la sección 4.2.1. Los códigos de Goppa encajados, ademas de haber una gran cantidad de ellos, son fáciles de conseguir, ya que para obtenerlos, según la nota 4.3, solo hay que estudiar la divisibilidad entre los polinomios de Goppa de los distintos códigos.

Ahora obtenemos una matriz generatriz, G_i, de cada uno de los M códigos de Goppa. Generamos una matriz de permutación aleatoria cuadrada de tamaño n, π, la cual emplearemos para permutar las n columnas de todas las matrices generatrices.

También generamos para cada matriz generatriz G_i, una matriz invertible aleatoria cuadrada de tamaño k_i, S_i, donde k_i hace referencia a la dimensión del código de Goppa C_i.

A partir de las matrices G_i, S_i y π obtenemos la clave pública, que constará de la matriz A, las M matrices obtenidas al enmascarar las matrices G_i, es decir, las matrices $G_i' = S_i \cdot G_i \cdot \pi$, con $1 \leq i \leq M$ y la capacidad correctora del código de producto de matrices C, t.

En cuanto a la clave privada, estará formada por la matriz de permutación π, las M matrices generatrices, las M matrices invertibles S_i y el algoritmo de descodificación del código de producto de matrices, ϑ, por tanto, consta de los algoritmos de descodificación de los M subcodigos.

En resumen, tenemos que las claves pública y privada, denotadas respectivamente por κ_p y κ_s están constituidas por:

$$\kappa_p = (A,\, G_1',\, ,\cdots,\, G_M', t) \text{ y } \kappa_s = (G_1, \cdots,\, G_M,\, \pi,\, S_1, \cdots,\, S_M,\, \vartheta)$$

Por tanto, podemos observar que el tamaño de dichas claves es inferior en cuanto trabajemos con códigos con parámetros elevados al tener que almacenar matrices de tamaño inferior respecto a la matriz generatriz enmascarada, G', del código de producto de matrices y las matrices S y P que es necesario emplear en el criptosistema original de McEliece.

9.2.2. Cifrado

Para realizar el cifrado de un mensaje empleando dicho criptosistema tenemos que emplear las distintas matrices que nos proporcionan a partir de la clave pública para generar una matriz que se corresponde con la matriz generatriz enmascarada G'.

Para ello simplemente construimos la matriz de igual forma que se realizaba la construcción de la matriz generatriz de los códigos producto de matrices visto en la nota 8.2

$$\begin{pmatrix} G'_1 & 0 & \cdots & 0 \\ 0 & G'_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G'_M \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{pmatrix} =$$

$$\begin{pmatrix} G_1 a_{11} & G_1 a_{12} & \cdots & G_1 a_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ G_M a_{M1} & G_M a_{M2} & \cdots & G_M a_{MM} \end{pmatrix} = G'$$

donde $A = (a_{ij})_{1 \leq i,j \leq M}$ y hemos empleado un abuso de notación al realizar una multiplicación de un elemento perteneciente a $\mathbb{F}_2$, a_{ij}, por una matriz, G_i.

Entonces, una vez obtenida la matriz G' simplemente multiplicamos el mensaje por dicha matriz para obtener su codificación, es decir, $c = m \cdot G'$.

Ahora a partir de la capacidad correctora de errores del código producto de matrices empleado, t, introducimos un vector de errores e de igual longitud que el vector c y con peso de Hamming inferior o igual a t. Por tanto obtenemos el mensaje cifrado tras sumar dicho vector de errores con el mensaje codificado: $y = c + e$.

9.2.3. Descifrado

Para realizar el descifrado del vector y, empleamos las matrices de la clave privada para generar una matriz de permutación cuadrada, Π, de tamaño $n * M$, similar a la empleada por Gaborit en la versión vista en la sección 7.2.3, y una matriz invertible S, las cuales se construyen como se muestra a continuación:

$$\Pi = \begin{pmatrix} \pi & 0 & \cdots & 0 \\ 0 & \pi & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \pi \end{pmatrix}, \quad S = \begin{pmatrix} S_1 & 0 & \cdots & 0 \\ 0 & S_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S_M \end{pmatrix},$$

donde π es la matriz de permutación cuadrada de tamaño n y S_i es la matriz invertible cuadrada de tamaño k_i.

A partir de estas matrices podemos realizar ya la descodificación de forma semejante a la versión original presentada por McEliece, que vimos en la sección 5.1.3, es decir, multiplicando el mensaje cifrado por la matriz de permutación inversa, Π^{-1}, obteniendo que $y \cdot \Pi^{-1} = m \cdot S \cdot G + e \cdot \Pi^{-1}$.

Como $e \cdot \Pi^{-1}$ sigue siendo un vector de peso de Hamming inferior o igual a t y además tenemos impuestas todas las condiciones necesarias para poder emplear el algoritmo de descodificación de los códigos de producto de matrices visto en la sección 8.2, aplicamos dicho algoritmo de descodificación y obtenemos $m \cdot S \cdot G$. Ahora realizamos la descodificación obteniendo el vector $m \cdot S$ y aplicando la matriz inversa de S sobre dicho vector recuperamos el mensaje.

De esta forma, hemos mostrado una versión del criptosistema de McEliece que reduce notablemente el tamaño de las claves pública y privada, manteniendo la seguridad de dicho criptosistema. Por tanto, se elimina de esta forma la principal desventaja que presentaba el criptosistema de McEliece original.

En cuanto a los códigos empleados para la construcción del código de producto de matrices, se expone en esta versión el empleo de códigos de Goppa binarios ya que, ademas de la facilidad de obtener códigos de Goppa binarios encajados y la gran cantidad que existe de ellos, presentan una buena seguridad, al ser códigos que todavía no han sido criptoanalizados exitosamente con la version original del criptosistema de McEliece. Sin embargo, es posible emplear cualquier otro tipo de códigos con el nuevo criptosistema presentado siempre que podamos garantizar, en cierta medida, que la seguridad no resulta comprometida al emplearlos.

Por último, se recomienda escoger los códigos de Goppa de forma que se maximice la distancia del código producto de matrices vista en la proposición 8.12, y que los polinomios que generan dichos códigos de Goppa sean separables, para que de esa forma, la capacidad correctora de dicho código sea lo mas grande posible por lo visto en la proposición 4.8.

Vamos a mostrar a continuación un ejemplo para el cual implementaremos el nuevo criptosistema de McEliece y lo compararemos con un criptosistema de McEliece que emplee un código de Goppa binario para su construcción.

Ejemplo 9.1. Para este ejemplo emplearemos la construcción de Plotkin como código de producto de matrices. Por tanto, tenemos que la matriz A es:

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

En cuanto a los códigos que utilizamos para la construcción del código producto de matrices, tomamos dos códigos de Goppa, ambos de longitud $2^5 = 32$, donde el vector L esta formado por todos los elementos del cuerpo $\mathbb{F}_{2^5}$ en ambos casos. Junto con el vector L, tenemos que el código C_1 se forma a partir del polinomio de grado 3, $g_1(x) = x^3 + (\alpha^4 + \alpha^2 + 1)x + \alpha^3 + \alpha^2 + \alpha + 1$ y C_2 a partir del polinomio de grado 6, $g_2(x) = x^6 + (\alpha^3 + \alpha^2)x^4 + (\alpha^2 + \alpha)x^3 + (\alpha^4 + \alpha^3 + \alpha + 1)x^2 + (\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1)x + \alpha^4 + \alpha^3$, siendo α un elemento primitivo de $\mathbb{F}_{2^5}$.

Ahora formamos el código de producto de matrices, C, a partir de los códigos C_1 y C_2 y la matriz A. Entonces, el código C tiene como parámetros fundamentales $[64, 20, 13]$ ya que los parámetros fundamentales de C_1 y C_2 son respectivamente $[32, 17, 7]$ y $[32, 3, 13]$

Como $g_1|g_2$ entonces se cumple que $C_2 \subset C_1$ y por tanto podemos emplear el algoritmo de descodificación visto en la sección 8.2. Obviamente, la matriz generatriz del código C_1, G_1, tiene tamaño 17×32 y la matriz generatriz del código C_2, G_2, tiene tamaño 3×32, luego la matriz generatriz, G, de C tiene tamaño 20×64.

Generamos ahora una matriz de permutación, π, cuadrada de tamaño 32 y dos matrices invertibles cuadradas S_1 y S_2 de tamaño 17 y 3 respectivamente.

Entonces, a partir de todas las matrices anteriores, podemos construir las matrices $G_1' = S_1 \cdot G_1 \cdot \pi$ y $G_2' = S_2 \cdot G_2 \cdot \pi$.

Con ellas, tenemos todas las matrices que, junto con la capacidad correctora de C, que es $t = 6$, y la matriz A nos permiten obtener la clave pública. De igual forma tenemos tanto las matrices como el algoritmo de descodificación necesarios para la clave privada.

Ahora vamos a realizar una comparación entre el criptosistema de McEliece que acabamos de construir y otro construido empleando la versión clásica explicada en la sección 5.1.

Para ello emplearemos un código de Goppa binario, C', cuyos parámetros fundamentales sean semejantes a los que tiene el código de producto de matrices empleado, para ver que, empleando códigos con longitudes y dimensiones semejantes, y por tanto con una redundancia del código, $\frac{k}{n}$, similar, podemos construir un criptosistema de McEliece dotado de una seguridad semejante y con una importante reducción del tamaño de las claves pública y privada que tenemos que almacenar.

Por tanto, consideramos C' con longitud $2^6 = 64$, luego el vector L esta formado por todos los elementos de $\mathbb{F}_{2^6}$. En cuanto al polinomio empleado para la construcción de C', empleamos un polinomio irreducible de grado 7: $g(x) = x^7 + \alpha^5 + \alpha^3 + \alpha$, donde α es un elemento primitivo de $\mathbb{F}_{2^6}$. De esta forma, obtenemos el código C' con dimensión $k' = 22 \approx 20 = k$ y distancia mínima $d' = 15 \approx 13 = d$.

Entonces, tenemos una matriz generatriz, G', de tamaño 22×64, luego necesitamos una matriz de permutación, P', cuadrada de tamaño 64 y una matriz invertible, S', cuadrada de tamaño 22 para enmascarar la matriz G'.

Por tanto, podemos observar que se reduce el tamaño de las claves pública y privada que tenemos que almacenar, ya que, de tener que guardar en la clave pública, la

matriz generatriz enmascarada del código C' de tamaño 22×64 en este caso, tenemos que almacenar una matriz de tamaño 17×32 y otra de tamaño 3×32 correspondientes a las matrices generatrices enmascaradas de los códigos C_1 y C_2, junto con la matriz A de tamaño 2×2.

Entonces, realizando la construcción del criptosistema de McEliece empleando el código de Goppa binario C', tendríamos que almacenar un total de 1408 bits o 176 bytes, mientras que empleando el código de producto de matrices para la construcción del nuevo criptosistema explicado tenemos que almacenar un total de $544 + 96 + 4 = 644$ bits o $\lfloor 80{,}5 \rfloor = 81$ bytes, lo cual reduce prácticamente a la mitad el tamaño de la clave pública que tenemos que transmitir y almacenar.

Ademas, vemos que estos resultados se obtienen a partir de un ejemplo que emplea una matriz A cuadrada de tamaño 2, es decir, que nuestro código de producto de matrices empleado se construye únicamente a partir de 2 códigos. Si empleamos en su lugar una matriz A de mayor tamaño, de forma que el código de producto de matrices que se emplee se construya a partir de una mayor cantidad de códigos, obtendremos una clave pública total de gran tamaño, mientras que las matrices generatrices enmascaradas correspondientes a los códigos empleados para la construcción del código de producto de matrices serán bastante mas pequeñas en comparación con la matriz generatriz enmascarada total. Por tanto, para esos casos la reducción del tamaño es mayor que para el ejemplo que hemos considerado.

En cuanto a la clave privada, si empleamos para la construcción del criptosistema el código de Goppa binario, C', tenemos que almacenar las matrices G', S' y P' junto con el algoritmo de descodificación del código.

Para la matriz G', el tamaño se reduce de igual forma que en la clave pública al corresponderse con matrices de iguales tamaños, es decir, 1408 bits con la construcción del criptosistema mediante el código de Goppa y 644 bits con la construcción mediante el código de producto de matrices.

Por último, vemos que las matrices S' y P' ocupan un espacio de 484 bits y 4096 bits respectivamente, mientras que las matrices S_1 y S_2 ocupan un espacio total de 298 bits, y la matriz π ocupa 1024 bits.

Debido a esto, vemos que se produce una reducción de los tamaños que ocupan todas las matrices que debemos almacenar a prácticamente la mitad, lo cual nos permite emplear el criptosistema de McEliece evitando, como ya mencionamos, su principal inconveniente.

Nota 9.2. Respecto a la nueva versión del criptosistema de McEliece presentada en este capítulo, podemos observar que para los códigos de producto de matrices, la matriz

A y los subcódigos empleados se deben encontrar en el mismo cuerpo. Por tanto, actualmente solo es posible realizar la construcción del nuevo criptosistema empleando la matriz correspondiente a la construcción de Plotkin, ya que vimos en el ejemplo 8.2 que no existe ninguna matriz cuadrada de tamaño 3 o superior formada únicamente por elementos de $\mathbb{F}_2$ que sea no singular por columnas.

Aunque para esta construcción hemos visto que se reduce notablemente el tamaño de las claves del criptosistema, habría que analizar la reducción de las claves ocasionada empleando el nuevo método a partir de códigos de producto de matrices que se construyan con subcodigos de Goppa sobre cuerpos no binarios y una matriz A de mayor tamaño, donde prevemos, como hemos dicho en el ejemplo 9.1, que dicha reducción será aun mayor que para la construcción de Plotkin. Sin embargo, habría que analizar si merece la pena incrementar mas la reducción de las claves, ya que al tener que emplear códigos de Goppa no binarios, se pierde la mitad de la capacidad correctora del código como vimos en la proposición 4.8.

Otro aspecto importante que faltaría por analizar detenidamente, es si la seguridad del criptosistema de McEliece se ha visto mermada mediante el empleo del nuevo método desarrollado. Para ello, habría que realizar un estudio comparativo acerca de todos los posibles ataques conocidos hasta el momento contra el criptosistema de McEliece, así como los ataques nuevos que podrían surgir al criptosistema debido al empleo de los códigos de producto de matrices.

Bibliografía

[1] DAVID MORENO CENTENO , *Implementación de la teoría de códigos: McEliece y una nueva versión*, Trabajo fin de grado informática, 2019.

[2] SHAMIR A, *A polynomial time algoritm for breaking the basic Merckle-Hellman cryptosystems*, IEEE trans on inform 1984.

[3] MERKLE, R. C., *A digital signature based on a conventional encryption function*, Conference on the Theory and Application of Cryptographic Techniques 1987.

[4] NICOLAS COURTOIS, ALEXANDER KLIMOV, JACQUES PATARIN Y ADI SHAMIR, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, Advances in cryptology—EUROCRYPT 2000.

[5] JUSTESEN JØRN, HØHOLDT TOM., *A Course in Error-Correcting Codes*, European Mathematical Society Publishing House 2004.

[6] YUAN XING LI, ROBERT H. DENG, AND XIN MEI WANG, *On the equivalence of McElieces and Niederreiters public-key cryptosystems*, IEEE Transactions on Information Theory 1994.

[7] NICOLAS COURTOIS, MATTHIEU FINIASZ, AND NICOLAS SENDRIER, *How to achieve a mceliece-based digital signature scheme*, Advances in Cryptology, ASIACRYPT 2001.

[8] V. M. SIDELNIKOV AND S. O. SHESTAKOV, *On the insecurity of cryptosystems based on generalized Reed-Solomon codes*, Discrete Math. Appl 1992.

[9] FAUGÈRE, J., GAUTHIER-UMANA, V., OTMANI, A., PERRET, L., TILLICH, J., *A distinguisher for high rate McEliece cryptosystems*, Proceedings IEEE Information Theory Workshop 2011.

[10] P. GABORIT, *Shorter keys for code based cryptography*, International Workshop on Coding and Cryptography 2005.

[11] SAN LING, *On the algebraic structure of quasi-cyclic codes .I. Finite fields*, Proceedings IEEE Information Theory Workshop 2001.

[12] T.A. GULLIVER, V.K. BHARGAVA, *A binary quasi-cyclic code*, Appl. Math. Lett. 1995.

[13] KRISTINE LALLY Y PATRICK FITZPATRICK, *Algebraic structure of quasicyclic codes*, Workshop on Coding and Cryptography 1999.

[14] JACQUES STERN, *A method for finding codewords of small weight*, Lecture Notes in Computer Science 1989.

[15] A. OTMANI, J.P. TILLICH, AND L. DALLOT, *Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes*, preprint 2008.

[16] T. P. BERGER, P. CAYREL, P GABORIT, AND A. OTMANI, *Reducing Key Length of the McEliece Cryptosystem*, AFRICACRYPT 2009.

[17] J.-C. FAUGÈRE, A. OTMANI, L. PERRET, AND J.-P. TILLICH, *Algebraic cryptanalysis of McEliece variants with compact keys*, Proc. Adv. Cryptol. 2010.

[18] V. G. UMAÑA AND G. LEANDER, *Practical key recovery attacks on two McEliece variants*, Association Cryptol. Res. (IACR) 2009.

[19] J.-C. FAUGÈRE, A. OTMANI, L. PERRET, F. DE PORTZAMPARC, AND J.-P. TILLICH, *Structural cryptanalysis of McEliece schemes with compact keys*, Designs codes Cryptography 2016.

[20] J.-C. FAUGÈRE, A. OTMANI, L. PERRET, F. DE PORTZAMPARC, AND J.-P. TILLICH, *Folding alternant and Goppa Codes with non-trivial automorphism groups*, IEEE Trans. Inf. Theory 2016.

[21] BLACKMORE, T., NORTON, G.H., *Matrix-product codes over* $\mathbb{F}_q$, Appl. Algebra Eng. Commun. Comput 2001.

[22] MACWILLIAMS, F.J., SLOANE, N.J.A., *The Theory of Error-Correcting Codes*, North-Holland Mathematical Library 1977.

[23] J. GATHEN, AND J. GERHARD, *Modern computer algebra*, Cambridge University 2013.

[24] —, *Algebraic Coding Theory*, New York: McGraw-Hill 1968.

[25] SHOUP AND, VICTOR, *A Computational Introduction to Number Theory and Algebra*, Cambridge University 2008.

[26] E. R. BERLEKAMP, *Goppa codes*, IEEE Trans. Inform. Theory 1973.

[27] R. PELLIKAAN, X. WU, S. BULYGIN AND, R. JURRIUS, *Codes, Cryptology and Curves with computer algebra*, Cambridge University 2018.

[28] R. THOMAS, *How sage helps to implement Goppa codes and McEliece PKCSs*, 2011.

[29] D. J. BERNSTEIN, *List decoding for binary Goppa codes*, Third International Workshop 2011.

[30] *Post-Quantum Cryptography | CSRC*, Computer Security division, Information Technology Laboratory, National Institute of Standars and Technology,.

URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`

[31] *Página web Indeed*

URL: `https://www.indeed.es/salaries/`

[32] A. COUVREUR, P. GABORIT, V. GAUTHIER-UMAÑA, A. OTMANI Y, J-P. TILLICH, *Distinguisher-Based Attacks on Public-Key Cryptosystems Using Reed-Solomon Codes*, 2014.

[33] I. MÁRQUEZ-CORBELLA, N. SENDRIER Y, M. FINIASZ, *Attack against GRS codes*, France université numérique, FUN-MOOC 2013.

URL:`https://www.fun-mooc.fr/courses/course-v1:inria+41006+archiveouvert/about`

[34] A. COUVREUR, A. OTMANI Y, J-P. TILLICH, *POLYNOMIAL TIME ATTACK ON WILD MCELIECE OVER QUADRATIC EXTENSIONS*, IEEE Transactions on Information Theory 2016.

[35] C. MUNUERA Y, J. TENA, *Codificación de la Información*, Universidad de Valladolid 1997.

[36] P. J. LEE Y, E. F. BRICKELL, *An observation on the security of McEliece's public-key cryptosystem*, Advances in cryptography- EUROCRYPT 88 1988.