Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

Criptosistema de McEliece

Autora: Lucía Fraile de Antonio
Tutor: Umberto Martínez Peñas
Año: 2022 - 2023

Índice general

Resumen

El criptosistema de McEliece es un criptosistema de clave pública basado en codificar un mensaje secreto utilizando como clave pública una matriz generadora de un código corrector de errores y añadiendo un error aleatorio. Dicha matriz generadora se construye multiplicando por una matriz S invertible por la izquierda y una matriz P de permutación por la derecha a otra matriz generadora de estructura conocida. Las matrices S y P se escogen al azar y constituyen la clave privada. Sin ellas, la estructura del código es difícil de adivinar (el código parece escogido al azar), por lo que corregir el error es difícil para un observador no deseado. Con la clave privada, podemos obtener la matriz generadora original, la cual descubre la estructura del código y para la cual se conoce un algoritmo eficiente de corrección de errores. Mediante dicho algoritmo corrector, podemos obtener el mensaje original. Este criptosistema no se utiliza en la actualidad debido al gran tamaño de sus claves. Sin embargo, se ha demostrado que puede resistir ataques que utilizan computación cuántica, en contraste con los criptosistemas actuales, por lo que es un buen candidato para la criptografía post-cuántica.

Abstract

McEliece's cryptosystem is a public key cryptosystem based on encoding a secret message using a generator matrix of an error-correcting code as public key and adding a random error. This generator matrix is constructed by multiplying an invertible matrix S on the left and a permutation matrix P on the right to another generator matrix of known structure. The matrixes S and P are chosen randomly and they constitute the private key. Without them, the code structure is difficult to guess (the code seems randomly chosen), so correcting the error is difficult for an unwanted observer. With the private key, we can obtain the original generator matrix, which discovers the code structure and for which an efficient error correcting algorithm is known. Using the said correction algorithm, we can get the original message. This cryptosystem is not currently been used due to the large size of its keys. However, it has been shown to withstand attacks using quantum computing, in contrast to current cryptosystems, making it a good candidate to post-quantum cryptography.

Introducción

En la actualidad, los procesos de transmisión de la información digital tienen una importancia primordial, ya que todos necesitamos enviar y recibir datos de manera segura a través de un canal poco fiable.

Los códigos criptográficos garantizan la seguridad del mensaje a enviar codificándolo, y después este se transmite por medio de un criptosistema que puede ser simétrico o asimétrico. Además, hay veces que durante la transmisión se han producido errores en la información enviada. Los códigos correctores, sirven para subsanar tales perturbaciones y obtener el mensaje original. Normalmente, la información codificada se transmite por medio de un criptosistema simétrico, y los criptosistemas asimétricos se utilizan para enviar la clave del criptosistema simétrico, que nos permitirá descodificar la información para obtener el mensaje original.

El objetivo principal de este trabajo es presentar el criptosistema de McEliece, un criptosistema asimétrico o de clave pública que en 1978 propuso R.J. McEliece en [5], construido a partir de un código corrector de errores con un algoritmo de descodificación eficiente. A pesar de que este criptosistema no se utilice en la práctica debido al gran tamaño de sus claves, veremos que tiene un gran interés ya que es uno de los principales candidatos a resistir ataques con ordenadores cuánticos. Esto es de gran importancia ya que los criptosistemas de clave pública utilizados actualmente, principalmente el RSA y ElGamal, pueden romperse utilizando ordenadores cuánticos.

En el primer capítulo de este trabajo se introducirán las nociones y los resultados sobre códigos criptográficos, códigos correctores y códigos lineales que necesitaremos para la elaboración del mismo.

Seguidamente, en el segundo capítulo, daremos la descripción, codificación y descodificación de tres códigos lineales: los Reed-Solomon, los alternantes y los Goppa, viendo además las relaciones que existen entre ellos. McEliece propuso los códigos Goppa para la descripción de su criptosistema, y aunque se han utilizado otros códigos para su implementación, los Goppa han resultado ser la única opción segura hasta el momento.

A continuación, abordamos en el tercer capítulo, el tema principal de este trabajo, el criptosistema de McEliece. Primero veremos qué son los problemas NP-completos

y después se describirá el criptosistema, observando que las ideas en las que se basa McEliece son problemas de tipo NP-completos, por lo que en un principio la intuición nos hace pensar que el criptosistema será seguro. Por último se expondrá un ejemplo académico de códigos Goppa, que nos servirá tanto para ver las nociones introducidas en el primer capítulo, como para describir un ejemplo del criptosistema de McEliece.

Después, se analizan en el último capítulo, los dos tipos de ataques que hay contra el criptosistema de McEliece: los ataques de descodificación genéricos y los ataques estructurales, viendo un ejemplo de algoritmo para cada uno de los ataques. De aquí, obtendremos propiedades que harán que el criptosistema sea más o menos seguro, dependiendo de si se utiliza un código u otro para su descripción.

Por último, terminaremos el trabajo señalando las conclusiones a las que hemos llegado después del estudio realizado sobre el criptosistema de McEliece.

La elaboración de este trabajo se apoya principalmente en los contenidos y resultados que se abordan en [4], [6] y [9], así como en [2] para la descripción de los códigos Reed-Solomon. Además, se han consultado los artículos [5] y [10] para la descripción del criptosistema de McEliece y los problemas NP-completos, respectivamente. También se hizo uso de [3] para describir un algoritmo que hace eficiente un ataque de descodificación de un código genérico, y [8] para el desarrollo de un ataque estructural propuesto contra el criptosistema utilizando códigos Reed-Solomon generalizados, además de [1] y [7] para los ataques con ordenadores cuánticos.

Capítulo 1

Preliminares de Códigos

En este primer capítulo introduciremos algunas definiciones y resultados básicos de la Teoría de la Información, así como de códigos, que se repetirán y tendrán relevancia a lo largo del trabajo.

Se entiende por transmisión de la información al proceso mediante el cual un emisor envía a un receptor un mensaje a través de un canal. Para ello, el emisor codifica la información y después la envía por el canal. Después de salir del canal de transmisión, dicha información debe ser descodificada para que el receptor reciba el mensaje original enviado no codificado. El esquema es el siguiente:

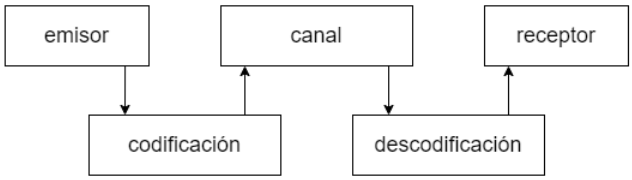

Figura 1.1: Esquema del proceso de transmisión de la información.

Un código criptográfico es aquel que codifica los mensajes a enviar, de manera que garantiza la privacidad de la información enviada. Así, si Alice quiere enviar un mensaje a Bob de forma segura, evitando que Eve conozca el mensaje, suplante la identidad, o modifique el mensaje, usará un sistema criptográfico, cuyo esquema es el de la figura 1.2.

Muchas veces, los canales de transmisión tienen ruido (es decir, interferencias o perturbaciones), por lo que se pueden producir errores en el mensaje codificado después de salir del canal. Un código corrector de errores es aquel que añade redundancia en el proceso de codificación, para poder corregir los errores provocados por el ruido al descodificar el mensaje enviado. El esquema del proceso de transmisión

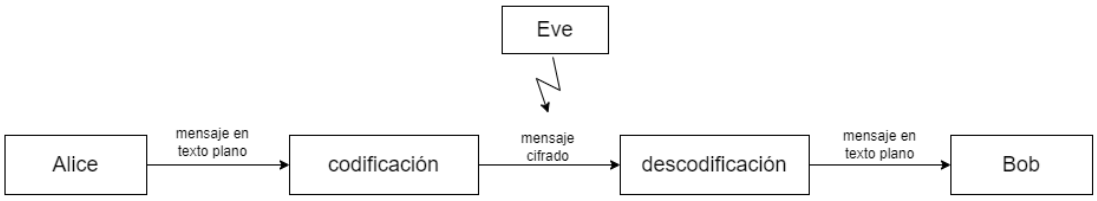

Figura 1.2: Esquema de un sistema criptográfico.

de la información por medio de un canal con ruido con su posterior corrección de errores se representa en la figura 1.3.

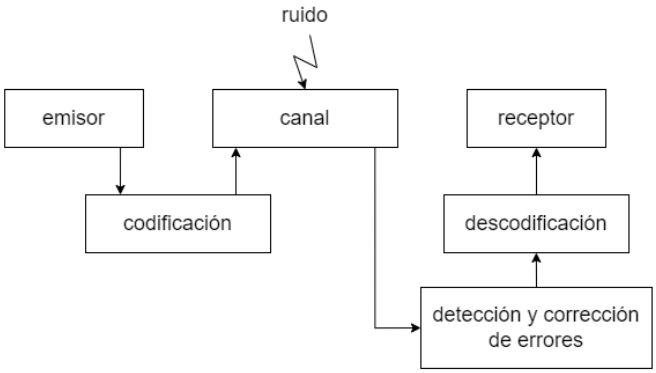

Figura 1.3: Esquema del proceso de transmisión de la información por medio de un canal con ruido y posterior corrección de errores.

Además, existen otro tipo de códigos, los códigos compresores, cuyo objetivo es comprimir la información para facilitar su transmisión, pero estos últimos no se abordarán, puesto que el objetivo principal del trabajo es el estudio del criptosistema de McEliece, que es un código criptográfico basado en la teoría de códigos correctores.

En la siguiente sección se introducirán los códigos criptográficos y después estudiaremos los códigos correctores.

1.1. Códigos Criptográficos

Como hemos visto anteriormente, tenemos el siguiente escenario: Alice quiere enviar un mensaje a Bob por medio de un canal de transmisión. Pero a su vez, Eve quiere conocer el mensaje.

La criptografía es el estudio del cifrado de mensajes. Consiste en diseñar e implementar canales de transmisión seguros, resistentes a ataques.

El criptoanálisis es el estudio del descifrado de mensajes. Trata de romper los canales de transmisión, aplicando distintos métodos.

La criptología es la ciencia que estudia el cifrado y descifrado de mensajes, es decir, engloba a la criptografía y al criptoanálisis.

Definición 1.1.1. Un criptosistema es un canal de transmisión que está formado por una terna $(\mathcal{M}, \mathcal{C}, \mathcal{K})$, donde:

$\mathcal{M}$ es el conjunto de mensajes originales o en texto plano;

$\mathcal{C}$ es el conjunto de mensajes cifrados;

$\mathcal{K} = (\mathcal{K}_1, \mathcal{K}_2)$ es el conjunto de pares de claves;

junto con una aplicación de cifrado $c : \mathcal{M} \times \mathcal{K}_1 \longrightarrow \mathcal{C}$ y una aplicación de descifrado $d : \mathcal{C} \times \mathcal{K}_2 \longrightarrow \mathcal{M}$, tales que

$$d(c(m, k), k') = m, \quad \forall m \in \mathcal{M}, \quad \forall (k, k') \in \mathcal{K}.$$

Supongamos que Alice conoce la clave k y Bob conoce la clave k' tal que $(k, k') \in \mathcal{K}$. Alice codifica el mensaje original $m \in \mathcal{M}$ que quiere enviar a Bob, con la clave k y la aplicación de cifrado c. Bob recibe $c(m, k)$, y para obtener el mensaje original, lo descodifica usando la clave k' y la aplicación de descifrado d, obteniendo $d(c(m, k), k') = m$.

Nota. Los elementos de $\mathcal{M}$ y $\mathcal{C}$ son sucesiones finitas de elementos de $\mathcal{A}$ y $\mathcal{B}$ respectivamente, donde $\mathcal{A}$ y $\mathcal{B}$ son alfabetos finitos (es decir, conjuntos finitos) que pueden ser o no iguales.

Se dice que un criptosistema es seguro si no se consigue descifrar el mensaje en un tiempo razonable, a menos que se conozca la clave k'.

A continuación abordaremos dos tipos de criptosistemas, los criptosistemas de clave privada y los criptosistemas de clave pública. En ambos, las aplicaciones de cifrado, $c : \mathcal{M} \times \mathcal{K}_1 \longrightarrow \mathcal{C}$, y descifrado, $d : \mathcal{C} \times \mathcal{K}_2 \longrightarrow \mathcal{M}$, se pueden considerar públicas, es decir, conocidas por cualquier usuario.

Definición 1.1.2. Un criptosistema simétrico o de clave privada es aquel donde todos los usuarios (normalmente pocos, idealmente dos) comparten y guardan en secreto las claves $k, k' \in \mathcal{K}$ de cifrado y descifrado (que pueden ser iguales).

Se denominan simétricos porque a partir de una de las claves es computacionalmente sencillo obtener la otra. Por ello, la seguridad de estos criptosistemas reside en mantener en secreto ambas claves.

El problema está en cómo los usuarios pueden compartir dichas claves. Esto se solventa, por ejemplo, utilizando criptosistemas de clave pública.

Definición 1.1.3. Se dice que un criptosistema es asimétrico o de clave pública si cada usuario tiene un par de claves $(k_{pub}, k_{priv}) \in \mathcal{K}$, tales que $d(c(m, k_{pub}), k_{priv}) = m$, y $c(d(c, k_{priv}), k_{pub}) = c$, para todo $m \in \mathcal{M}$ y todo $c \in \mathcal{C}$.

La clave k_{pub} es pública, es decir, todos los usuarios tienen acceso a ella. Mientras que la clave k_{priv} es privada, es decir, es conocida solamente por el receptor.

Estos criptosistemas se denominan asimétricos porque se cifra con k_{pub}, se descifra con k_{priv}, y $k_{pub} \neq k_{priv}$. Por ello, la seguridad de estos criptosistemas reside en mantener en secreto k_{priv}.

De esta forma, si k_{pubB} y k_{privB} son las claves pública y privada de Bob, cualquier usuario puede enviar un mensaje $m \in \mathcal{M}$ a Bob haciendo: $c(m, k_{pubB})$, y Bob será el único que pueda descifrarlo haciendo: $d(c(m, k_{pubB}), k_{privB}) = m$.

Todos los criptosistemas de clave pública deben satisfacer las condiciones de Diffie-Hellman:

1. La generación de k_{pub} y k_{priv} debe ser computacionalmente sencilla.
2. El proceso de cifrado, conocida k_{pub}, debe ser computacionalmente sencillo.
3. El proceso de descifrado, conocida k_{priv}, debe ser computacionalmente sencillo.
4. Obtener el mensaje original m a partir del mensaje cifrado $c(m, k_{pub})$, k_{pub} y las aplicaciones de cifrado y descifrado, pero sin conocer k_{priv}, debe ser computacionalmente difícil (o imposible).
5. Obtener k_{priv} a partir de k_{pub} debe ser computacionalmente difícil (o imposible).

Nota. Cuando decimos computacionalmente sencillo nos referimos a que existen algoritmos con complejidad polinómica. Mientras que computacionalmente difícil quiere decir que solo se conocen algoritmos de complejidad exponencial.

En general, las aplicaciones de cifrado de un criptosistema de clave pública son funciones de una vía con trampa.

Definición 1.1.4. Sea $f : X \to Y$ una aplicación. Se dice que f es una función de una vía si:

i) calcular $f(x) \in Y$ es computacionalmente sencillo, para cada $x \in X$;

ii) dado $y \in Y$, calcular $x \in X$ tal que $y = f(x)$ es computacionalmente difícil.

Se dice que f es una función de una vía con trampa si es una función de una vía y además cumple que:

iii) existe una información adicional gracias a la cual es computacionalmente sencillo calcular $x \in X$ dado $y = f(x)$.

Aunque los criptosistemas asimétricos son más seguros al no tener que transmitir las claves, estos necesitan claves con un número de bits mucho mayor que las claves de los criptosistemas simétricos. Además los algoritmos de cifrado y descifrado de los criptosistemas asimétricos son más lentos. En la práctica, se utilizan ambos, los criptosistemas de clave pública para transmitir las claves del criptosistema de clave privada, y estos últimos para cifrar el mensaje en texto plano y enviarlo.

1.2. Códigos Correctores y Códigos Lineales

Los códigos correctores se usan para detectar y corregir los posibles errores que se han producido durante la transmisión por el canal con ruido. Sin embargo, nosotros los utilizaremos para construir funciones de una vía con trampa, para diseñar un criptosistema de clave pública, el de McEliece.

Cuando se trabaja con este tipo de códigos, es habitual referirse al proceso de detección y corrección de errores que aparece en la Figura 1.3 como descodificación. Como ya hemos dicho, la idea principal de estos códigos es introducir información redundante (dígitos de control) que nos ayude a detectar y corregir tales errores que se han producido durante la transmisión.

Definición 1.2.1. Codificar consiste en dar una aplicación inyectiva $c : \mathcal{A}^k \longrightarrow \mathcal{A}^n$.

Se denomina código en bloque a la imagen de la aplicación de codificación c, es decir, al subconjunto $\mathcal{C} = \mathrm{Im}(c) \subset \mathcal{A}^n$.

Los elementos de $\mathcal{C}$ se llaman palabras del código o palabras. Se denomina longitud de una palabra del código al número de símbolos que tiene. Así, todas las palabras de $\mathcal{C}$ tienen longitud n.

Nótese que los códigos en bloque tienen descodificación única dado que la aplicación de codificación es inyectiva. Por este motivo, los códigos en bloque se utilizan en la detección y corrección de errores.

Una forma de describir un código en bloque es dando todas las palabras que lo forman, lo cual supone un gran coste de almacenamiento. Además, los procesos de codificación y descodificación de los códigos en bloque son muy costosos computacionalmente. Esto condujo a buscar códigos con una estructura de espacio vectorial como los códigos lineales.

Nota. Consideraremos a lo largo del trabajo el cuerpo finito $\mathbb{F}_q$, donde $q = p^r$ con p un primo y r un entero positivo. A partir de ahora, el alfabeto será $\mathbb{F}_q$, y se utilizarán códigos lineales, salvo que se indique lo contrario.

Definición 1.2.2. Un código lineal $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ es un subespacio vectorial de $\mathbb{F}_q^n$, para algún entero $n \geq 1$.

Nótese que los códigos lineales de longitud n son códigos en bloque de longitud n.

Como $\mathcal{C}$ es un subespacio vectorial, podemos hablar de su dimensión sobre $\mathbb{F}_q$, $\dim(\mathcal{C}) = k$ (es decir, $\mathcal{C}$ tiene k palabras linealmente independientes).

La base de los códigos correctores es crear códigos cuyas palabras sean muy distintas, para que sea poco probable que las alteraciones producidas en una palabra enviada por el canal, den como resultado otra palabra del código. En la mayoría de situaciones, la noción que mide esto se denomina distancia de Hamming, aunque también hay otras distancias que se usan.

Definición 1.2.3. Sean $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$. Se denomina distancia de Hamming entre $\mathbf{x}$ e $\mathbf{y}$ al número de coordenadas distintas entre $\mathbf{x}$ e $\mathbf{y}$, es decir,

$$d(\mathbf{x}, \mathbf{y}) = \#\{i : 1 \leq i \leq n, x_i \neq y_i\}.$$

Proposición 1.2.1. La distancia de Hamming cumple las propiedades de una distancia, es decir, para cada $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$ se verifica que:
i) $d(\mathbf{x}, \mathbf{y}) \geq 0$
ii) $d(\mathbf{x}, \mathbf{y}) = 0$ si y solo si $\mathbf{x} = \mathbf{y}$
iii) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
iv) $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$ (*Desigualdad triangular*)

Demostración. Las tres primeras propiedades se deducen de la definición de distancia de Hamming.
Probemos la desigualdad triangular:

- Si $x_i = z_i$ para todo $i = 1, 2, \ldots, n$, entonces se tiene que $d(\mathbf{x}, \mathbf{z}) = 0$, y por (i) se deduce que $d(\mathbf{x}, \mathbf{z}) = 0 \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.
- Si existe $i \in \{1, 2, \ldots, n\}$ tal que $x_i \neq z_i$ entonces, $x_i \neq y_i$ o $y_i \neq z_i$, de donde se deduce la desigualdad buscada.

□

Definición 1.2.4. Se denomina distancia mínima del código $\mathcal{C}$ al mínimo de las distancias de Hamming entre palabras distintas del código, es decir,

$$d = d(\mathcal{C}) = \text{mín}\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\}.$$

Se dice que un código lineal $\mathcal{C}$ sobre $\mathbb{F}_q$ es de tipo $[n, k, d]$ si tiene longitud n, dimensión k y distancia mínima d. Si conocemos la longitud y la dimensión de un código, pero no su distancia mínima, entonces se dice que el código es de tipo $[n, k]$. Además, n, k, d se denominan parámetros fundamentales del código $\mathcal{C}$.

Definición 1.2.5. Sea $\mathcal{C}$ un código lineal sobre $\mathbb{F}_q$ de tipo $[n, k]$.

Se denomina redundancia del código a $r = n - k$ y tasa de transmisión de información a $\frac{k}{n} \in (0, 1)$.

Definición 1.2.6. Se define el peso de Hamming de $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ como el número de dígitos no nulos de $\mathbf{x}$, es decir,

$$w(\mathbf{x}) = \#\{i : 1 \leq i \leq n, x_i \neq 0\}.$$

Un buen código será aquel que tenga poca redundancia y altas capacidades de detección y corrección de errores. Estas últimas quedan determinadas por la distancia mínima.

En el caso de los códigos lineales, la distancia mínima coincide con el mínimo de los pesos de Hamming.

Lema 1.2.1. La distancia mínima de un código lineal $\mathcal{C}$ es el menor de los pesos de Hamming de una palabra no nula del código. Es decir, $d = \text{mín}\{w(\mathbf{c}) : \mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}\}$.

Demostración. Sean $\mathbf{x}, \mathbf{y} \in \mathcal{C} \subset \mathbb{F}_q^n$ distintos, entonces $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$. Nótese que tiene sentido considerar $w(\mathbf{x}-\mathbf{y})$ ya que por ser $\mathcal{C}$ un espacio vectorial, $\mathbf{x}-\mathbf{y} \in \mathcal{C}$. Por lo tanto, la distancia entre dos palabras del código es el peso de Hamming de otra palabra del código.

Además, por ser $\mathcal{C}$ espacio vectorial, $\mathbf{0} \in \mathcal{C}$ y se puede considerar $d(\mathbf{x}, \mathbf{0}) = w(\mathbf{x})$ para cada $\mathbf{x} \in \mathcal{C}$ no nulo. Por lo que también, el peso de Hamming de una palabra del código es la distancia entre dos palabras del código.

Se deduce por lo tanto que el mínimo de las distancias entre dos palabras del código (la distancia mínima) es el mínimo de los pesos de Hamming de las palabras del código. □

Teorema 1.2.1. Sea $\mathcal{C}$ un código lineal de tipo $[n, k, d]$. Entonces, $\mathcal{C}$ puede corregir t errores si y solo si $2t < d$.

Demostración. Supongamos que $t \geq \frac{d}{2}$, y veamos que entonces $\mathcal{C}$ no corrige todos los errores de tamaño t.

Por el lema anterior, existe $\mathbf{c} \in \mathcal{C}$ tal que $w(\mathbf{c}) = d$.

Se construye $\mathbf{x} \in \mathbb{F}_q^n$ cambiando $\lceil \frac{d}{2} \rceil$ de las d coordenadas no nulas de $\mathbf{c}$ a 0. Por lo tanto, $d(\mathbf{c}, \mathbf{x}) = \lceil \frac{d}{2} \rceil \leq t$ (pues estamos suponiendo que $t \geq \frac{d}{2}$).

Por otra parte, $d(\mathbf{0}, \mathbf{x}) \leq d - \lceil \frac{d}{2} \rceil \leq \frac{d}{2} \leq t$.

Así, se tiene que tanto la palabra $\mathbf{c}$ como $\mathbf{0}$ podrían haber sido obtenidas a partir de $\mathbf{x}$ al añadir un error de peso menor o igual que t. Por lo tanto, no se puede descodificar $\mathbf{x}$ de forma única, ya que tanto $\mathbf{c}$ como $\mathbf{0}$ serían candidatos adecuados.

Razonemos el recíproco por reducción a lo absurdo.

Supongamos que $t < \frac{d}{2}$, se envía $\mathbf{c} \in \mathcal{C}$ y se recibe $\mathbf{r} \in \mathbb{F}_q^n$ tal que se han producido a lo sumo t errores, es decir, con $d(\mathbf{c}, \mathbf{r}) \leq t$.

Si $\mathbf{x}$ es otra palabra del código a menor distancia de $\mathbf{r}$, es decir, tal que $d(\mathbf{r}, \mathbf{x}) \leq \frac{d-1}{2}$. Entonces, por la desigualdad triangular se deduce que:

$$d(\mathbf{c}, \mathbf{x}) \leq d(\mathbf{c}, \mathbf{r}) + d(\mathbf{r}, \mathbf{x}) \leq \frac{d-1}{2} + \frac{d-1}{2} = d - 1,$$

pero esto es absurdo ya que $\mathbf{c}, \mathbf{x} \in \mathcal{C}$ y d es la distancia mínima del código. Por tanto, $\mathbf{r}$ se puede descodificar de forma única como $\mathbf{c} \in \mathcal{C}$.

□

Definición 1.2.7. Sea d la distancia mínima del código lineal $\mathcal{C}$.

Se dice que $\mathcal{C}$ es t-corrector con $t = \lfloor \frac{d-1}{2} \rfloor$.

A continuación, se formalizará la idea de que si $\mathcal{C}$ es un código lineal de tipo $[n, k, d]$, podemos pensar en k de las n coordenadas de las palabras como la información que queremos enviar y las otras $n - k$ coordenadas como los dígitos de control que permitirán detectar y/o corregir los errores de transmisión.

1.2.1. Matriz Generadora

Sea $\mathcal{C}$ un código lineal sobre $\mathbb{F}_q$ de tipo $[n, k, d]$. Por ser $\mathcal{C}$ un subespacio vectorial de $\mathbb{F}_q^n$ de dimensión k, se tiene que existe una aplicación $f : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$ lineal e inyectiva cuya imagen es $\mathcal{C}$.

Definición 1.2.1.1. Se denomina matriz generadora o generatriz de $\mathcal{C}$ a la matriz de una aplicación $f : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$ lineal e inyectiva cuya imagen es $\mathcal{C}$, es decir, $G \in \mathfrak{M}_{k \times n}(\mathbb{F}_q)$ cuyas filas son una base de $\mathcal{C}$.

Nota. Existen varias aplicaciones f y para cada una de ellas existe una matriz generadora G. Se tienen entonces varias formas de expresar un mismo código.

Además, una matriz generadora proporciona una codificación, ya que todas las palabras del código pueden obtenerse haciendo combinaciones lineales de las filas de la matriz, es decir, multiplicando a la izquierda de la matriz por un vector fila. Así, un mensaje $\mathbf{x} \in \mathbb{F}_q^k$ se codifica por $\mathbf{x}G \in \mathbb{F}_q^n$, pues $\mathcal{C} = \{\mathbf{x}G : \mathbf{x} \in \mathbb{F}_q^k\}$, con complejidad cuadrática, $O(nk)$. De hecho, la aplicación que lleva $\mathbf{x}$ al vector $\mathbf{x}G$ es precisamente f.

De esta manera, para guardar el código $\mathcal{C}$, que tiene tamaño q^k, solo necesitaremos almacenar en memoria los nk elementos en $\mathbb{F}_q$ de la matriz G. Por el contrario, si tenemos un código en bloque no lineal de tamaño q^k, se requerirá guardar nq^k elementos.

Definición 1.2.1.2. Se dice que una matriz generadora G está en forma estándar si $G = (I_k|A)$, donde I_k denota a la matriz identidad de tamaño $k \times k$ y A es una matriz de tamaño $k \times (n-k)$ con coeficientes en $\mathbb{F}_q$, es decir, $A \in \mathfrak{M}_{k\times(n-k)}(\mathbb{F}_q)$.

Se dice que un código es sistemático si tiene una matriz generatriz en forma estándar.

Nótese que si G es una matriz generatriz en forma estándar, entonces para cada $\mathbf{x} \in \mathbb{F}_q^k$, $\quad \mathbf{x}G = \mathbf{x}(I_k|A) = (\mathbf{x}, \mathbf{z})$, y se tendrá que $\mathbf{z}$ son los dígitos de control. De ahí que se denominen códigos sistemáticos. Obsérvese, que además, si G es una matriz generatriz en forma estándar, la complejidad computacional de codificar es menor, $O(k(n-k))$.

Definición 1.2.1.3. Se dice que $M \in \mathfrak{M}_{n\times n}(\mathbb{F}_q)$ es una matriz monomial si todas sus filas y columnas tienen exactamente una entrada no nula. Equivalentemente, si P_σ es una matriz de permutación $n \times n$ y $\alpha_i \in \mathbb{F}_q \backslash \{0\}$ para todo $i = 1, \ldots, n$, entonces $M = P_\sigma \begin{pmatrix} \alpha_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_n \end{pmatrix}$ es una matriz monomial.

Sean $\mathcal{C}_1$ y $\mathcal{C}_2$ códigos lineales de longitud n sobre $\mathbb{F}_q$. Se dice que $\mathcal{C}_1$ y $\mathcal{C}_2$ son equivalentes si existe una matriz monomial M tal que $\mathcal{C}_2 = \{\mathbf{c}M : \mathbf{c} \in \mathcal{C}_1\}$.

Nótese que dos códigos equivalentes son del mismo tipo, es decir, tienen los mismos parámetros n, k, d.

Nota. No todo código lineal es sistemático, pero todo código lineal es equivalente a uno sistemático.

Hemos visto que podemos describir el subespacio vectorial $\mathcal{C}$ de $\mathbb{F}_q^n$ mediante una base, que da lugar a una matriz generadora. Pero también podemos expresarlo mediante unas ecuaciones implícitas, lo que dará lugar a una matriz de control.

1.2.2. Matriz de Control

Definición 1.2.2.1. Sea $\mathcal{C}$ un código lineal sobre $\mathbb{F}_q$ de tipo $[n, k, d]$. Se dice que $H \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_q)$ es una matriz de control de $\mathcal{C}$ si para todo $\mathbf{x} \in \mathbb{F}_q^n$ se cumple que:

$$\mathbf{x} \in \mathcal{C} \text{ si y solo si } H\mathbf{x}^t = \mathbf{0}^t.$$

Por lo tanto, una matriz de control permite detectar los elementos de $\mathbb{F}_q^n$ que están en $\mathcal{C}$ con complejidad polinómica, $O(n(n-k))$.

Nota. Si $H \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_q)$ es una matriz de control de $\mathcal{C}$, entonces tiene rango máximo, es decir, $n-k$:

Sea $(h_1, \ldots, h_n)$ una fila de la matriz de control y $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}$, entonces, por ser H matriz de control, se cumple que $h_1x_1 + \ldots + h_nx_n = 0$, de donde tenemos

una ecuación implícita de $\mathcal{C}$. Dado que se necesitan $n-k$ ecuaciones implícitas linealmente independientes para describir $\mathcal{C}$, H ha de tener $n-k$ filas linealmente independientes.

Observemos que al igual que con la matriz generadora, la matriz de control no es única, depende de la elección de las ecuaciones implícitas.

En la siguiente proposición se describe la relación que hay entre las matrices generatriz y de control de un código.

Proposición 1.2.2.1. Sean G y H las matrices generatriz y de control de un código, respectivamente. Entonces $GH^t = 0$.

Demostración. Como G es una matriz generatriz, sus filas son palabras del código. Además, por ser H matriz de control, se cumple que $HG^t = 0$, que es equivalente a que $GH^t = 0$. □

Proposición 1.2.2.2. Si la matriz generatriz viene dada en forma estándar, $G = (I_k|A)$, entonces, la matriz de control queda determinada por $H = (-A^t|I_{n-k})$.

Demostración. Si G y H tienen la forma del enunciado y se cumple que $HG^t = 0$, entonces, como G es por hipótesis una matriz generatriz del código, H será la matriz de control asociada.

$$HG^t = \begin{pmatrix} -A^t & | & I_{n-k} \end{pmatrix} \begin{pmatrix} I_k \\ \hline A^t \end{pmatrix} = -A^t + A^t = 0.$$

□

Veremos a continuación la relación que hay entre la distancia mínima y una matriz de control del código. Para ello necesitaremos el siguiente lema.

Lema 1.2.2.1. Sea $\mathcal{C}$ un código lineal de tipo $[n,k]$ con matriz de control H. Entonces, H tiene j columnas linealmente dependientes si y solo si existe $\mathbf{c} \in \mathcal{C}$ tal que su peso es $w(\mathbf{c}) \leq j$.

Demostración. Supongamos que H tiene j columnas linealmente dependientes, $\mathbf{h}_{i_1}$, $\ldots, \mathbf{h}_{i_j}$. Entonces, existen $a_{i_1}, \ldots, a_{i_j} \in \mathbb{F}_q$ no todos nulos tal que $a_{i_1}\mathbf{h}_{i_1} + \ldots + a_{i_j}\mathbf{h}_{i_j} = \mathbf{0}$. Completando a una n-upla con ceros en las correspondientes posiciones que no involucran a las columnas linealmente dependientes, obtenemos un vector $\mathbf{x} \in \mathbb{F}_q^n$ tal que $H\mathbf{x}^t = \mathbf{0}^t$, con lo que es una palabra del código y además tiene peso a lo sumo j, pues puede ser que algún a_{i_k} sea nulo para $k \in \{1, \ldots, j\}$, pero no todos.

Recíprocamente, sea $\mathbf{c} \in \mathcal{C}$ tal que $w(\mathbf{c}) \leq j$. Por ser H matriz de control, se tiene que $H\mathbf{c}^t = \mathbf{0}^t$, con lo que se tiene que j columnas de H son linealmente dependientes ya que el peso de $\mathbf{c}$ era a lo sumo j.

□

Proposición 1.2.2.3. Sea $\mathcal{C}$ un código lineal de tipo $[n, k]$ con matriz de control H. La distancia mínima de $\mathcal{C}$ es el menor número de columnas linealmente dependientes de H.

Demostración. Por el lema 1.2.1, $d = d(\mathcal{C}) = \text{mín}\{w(\mathbf{c}) : \mathbf{c} \in \mathcal{C}, \mathbf{c} \neq 0\}$. Y enlazándolo con el lema 1.2.2.1 anterior, se deduce que la distancia mínima del código es el menor número de columnas linealmente dependientes de la matriz de control. □

1.2.3. Código Dual

Sea $\mathcal{C}$ un código lineal con matriz de control H. Dado que el rango de H es máximo (por ser matriz de control), H se puede ver como una matriz generadora de otro código sobre $\mathbb{F}_q$, es decir, de otro código lineal.

Definición 1.2.3.1. Sea $\mathcal{C}$ un código lineal con matriz de control H. Se dice que $\mathcal{C}^\perp$ es el código dual de $\mathcal{C}$ si es un código lineal con H como matriz generatriz.

Proposición 1.2.3.1. Si $\mathcal{C}$ es un código lineal sobre $\mathbb{F}_q$ de tipo $[n, k]$ con G como matriz generadora y H matriz de control, entonces $\mathcal{C}^\perp$ es un código lineal sobre $\mathbb{F}_q$ de tipo $[n, n-k]$ con matriz generadora H y matriz de control G.

Demostración. Si $\mathcal{C}^\perp$ es el código dual de $\mathcal{C}$, entonces por definición se tiene que $H \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_q)$ es una matriz generadora de $\mathcal{C}^\perp$, por lo que $\mathcal{C}^\perp$ es de tipo $[n, n-k]$.

Además, por ser G y H matrices generadora y de control de $\mathcal{C}$, se cumple que $GH^t = 0$, que es equivalente a que $HG^t = 0$, de donde se deduce que G es una matriz de control de $\mathcal{C}^\perp$ por ser H una matriz generadora de $\mathcal{C}^\perp$. □

Corolario 1.2.3.1. Sea $\mathcal{C}$ un código lineal, entonces $(\mathcal{C}^\perp)^\perp = \mathcal{C}$.

Demostración. De la proposición anterior se deduce que $\mathcal{C}$ y $(\mathcal{C}^\perp)^\perp$ tienen las mismas matrices generadora y de control y por lo tanto los códigos han de ser iguales, ya que las matrices generatriz y de control definen un código. □

El siguiente corolario nos muestra una forma alternativa de definir el código dual $\mathcal{C}^\perp$ de $\mathcal{C}$, sin utilizar una matriz de control de $\mathcal{C}$.

Corolario 1.2.3.2. Sea $\mathcal{C}$ un código lineal, entonces las palabras de $\mathcal{C}^\perp$ son ortogonales a las de $\mathcal{C}$ para el producto escalar usual. Es decir, si $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}$ e $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{C}^\perp$, entonces $x_1y_1 + \ldots + x_ny_n = 0$. De hecho, $\mathcal{C}^\perp$ es el conjunto de las n-uplas que son ortogonales a todas las palabras de $\mathcal{C}$, es decir,

$$\mathcal{C}^\perp = \left\{ (y_1, \ldots, y_n) \in \mathbb{F}_q^n : \sum_{i=1}^{n} x_i y_i = 0, \forall (x_1, \ldots, x_n) \in \mathcal{C} \right\}.$$

Demostración. Sean $\mathbf{x} \in \mathcal{C}$, $\mathbf{y} \in \mathcal{C}^\perp$, H una matriz de control de $\mathcal{C}$ y $\mathbf{h}_i = (h_{i1}, \ldots, h_{in})$, $1 \leq i \leq n-k$ las filas de H.

Por una parte, como H es matriz de control de $\mathcal{C}$ y $\mathbf{x} \in \mathcal{C}$, se tiene que $H\mathbf{x}^t = \mathbf{0}^t$, es decir, $h_{i1}x_1 + \ldots + h_{in}x_n = 0, \quad 1 \leq i \leq n-k$.

Por otro lado, como H es matriz generadora de $\mathcal{C}^\perp$ e $\mathbf{y} \in \mathcal{C}^\perp$, existen $\alpha_1, \ldots, \alpha_{n-k} \in \mathbb{F}_q$ tal que $\mathbf{y} = \alpha_1\mathbf{h}_1 + \ldots + \alpha_{n-k}\mathbf{h}_{n-k}$.

Entonces,

$$\sum_{j=1}^{n} x_j y_j = \sum_{j=1}^{n} x_j \left(\sum_{i=1}^{n-k} \alpha_i h_{ij} \right) = \sum_{i=1}^{n-k} \alpha_i \left(\sum_{j=1}^{n} h_{ij} x_j \right) = \sum_{i=1}^{n-k} \alpha_i 0 = 0.$$

Así hemos probado que $\mathcal{C}^\perp \subset \left\{ (y_1, \ldots, y_n) \in \mathbb{F}_q^n : \sum_{i=1}^{n} x_i y_i = 0, \forall (x_1, \ldots, x_n) \in \mathcal{C} \right\}$.

Veamos que se tiene la otra contención.

Sea $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ tal que $\sum_{i=1}^{n} x_i y_i = 0$, para todo $(x_1, \ldots, x_n) \in \mathcal{C}$. Consideremos G una matriz generadora de $\mathcal{C}$ y $\mathbf{g}_i = (g_{i1}, \ldots, g_{in}), 1 \leq i \leq k$ las filas de G, que también son palabras de $\mathcal{C}$. Entonces, por hipótesis, $\sum_{j=1}^{n} g_{ij} y_j = 0$ para todo $i = 1, \ldots, k$ y como G también es matriz de control de $\mathcal{C}^\perp$, se concluye que $\mathbf{y} \in \mathcal{C}^\perp$. □

Definición 1.2.3.2. Se dice que $\mathcal{C}$ es un código autodual si cumple que $\mathcal{C}^\perp = \mathcal{C}$.

Nótese que si $\mathcal{C}$ es un código autodual, necesariamente n es un número par, pues ha de cumplirse que $n - k = k$.

Nota. Se ha visto que la longitud y la dimensión de un código lineal y su dual están relacionados. Sin embargo, a priori no se puede decir nada de la distancia mínima del dual de un código.

1.2.4. Descodificación de Códigos Lineales

Sea $\mathcal{C}$ un código lineal de tipo $[n, k, d]$ sobre $\mathbb{F}_q$.

Supongamos que se envía la palabra $\mathbf{c} \in \mathcal{C}$ y se recibe $\mathbf{r} \in \mathbb{F}_q^n$. Se tiene que $\mathbf{r} = \mathbf{c} + \mathbf{e}$, donde $\mathbf{e}$ es el error cometido. Si los errores se producen bit a bit, entonces $w(\mathbf{e})$ es el número de errores cometidos durante la transmisión.

Si el ruido del canal no es excesivo, el peso $w(\mathbf{e})$ no será muy alto, y por tanto se podrá corregir por el teorema 1.2.1. Una forma de descodificar $\mathbf{r}$ es con el algoritmo 1.

Algorithm 1: Algoritmo de descodificación por fuerza bruta

Input : La palabra recibida $\mathbf{r} \in \mathbb{F}_q^n$
Output: La palabra descodificada $\mathbf{c} \in \mathcal{C}$, o error

```
for each x ∈ C do
    d(r, x)
if there is only one c ∈ C with d(r, c) = mín{d(r, x) : x ∈ C} then
    return c
else
    return error
```

Dado que $\mathcal{C}$ es t-corrector, con $t = \lfloor \frac{d-1}{2} \rfloor$, si se ha obtenido una descodificación, podemos decir que:

- Si se han cometido como mucho t errores, es decir, si $d(\mathbf{c}, \mathbf{r}) = w(\mathbf{e}) \leq t$, entonces $\mathbf{c}$ es la única palabra del código con tal propiedad, por lo que la descodificación es correcta.
- Si $t < w(\mathbf{e})$, entonces la descodificación no es correcta, en general.

Aunque la descodificación obtenida a partir del algoritmo 1 sea correcta, el algoritmo es poco interesante porque tiene complejidad exponencial en k, $O(q^k)$, ya que en el peor caso habría que comprobar todas las palabras de $\mathcal{C}$ y por lo tanto, el número de operaciones puede llegar a ser un múltiplo de $|\mathcal{C}| = q^k$.

1.2.5. Cota de Singleton y Códigos MDS

La cota de Singleton nos dice que si un código tiene poca redundancia, entonces tendrá poca capacidad correctora, es decir, distancia mínima baja.

Teorema 1.2.5.1. Cota de Singleton.
Si $\mathcal{C}$ es un código lineal de tipo $[n, k, d]$, entonces $d \leq n - k + 1$.

Demostración. Sea H una matriz de control del código. Como la distancia mínima del código es el menor número de columnas linealmente dependientes de H y de estas hay a lo sumo $n - k + 1$, ya que el rango de H es $n - k$, se concluye que $d \leq n - k + 1$. □

Definición 1.2.5.1. Los códigos donde se da la igualdad en la cota de Singleton, es decir, aquellos tales que $d = n - k + 1$, se denominan códigos MDS (Maximun Distance Separable Codes).

Los códigos MDS son interesantes porque tienen la mayor distancia mínima posible, en virtud de la cota de Singleton.

Proposición 1.2.5.1. Si $\mathcal{C}$ es un código MDS, entonces su dual $\mathcal{C}^\perp$ es también un código MDS.

Demostración. Sea H una matriz de control de $\mathcal{C}$, luego también es una matriz generadora de $\mathcal{C}^\perp$.

Como $d = d(\mathcal{C})$ es el menor número de columnas linealmente dependientes de H y $\mathcal{C}$ es un código MDS, se deduce que H tiene $n-k$ columnas linealmente independientes.

Además, como $d^\perp = d(\mathcal{C}^\perp) = \min\{w(\mathbf{c}') : \mathbf{c}' \in \mathcal{C}^\perp, \mathbf{c}' \neq \mathbf{0}\}$, los elementos de $\mathcal{C}^\perp$ son de la forma $\mathbf{x}H$, con $\mathbf{x} \in \mathbb{F}_q^{n-k}$ y H tiene $n-k$ columnas linealmente independientes, se deduce que $d^\perp > n-(n-k) = k$.

Por otro lado, de la cota de Singleton tenemos que $d^\perp \leq n-(n-k)+1 = k+1$.

Se concluye así que $d^\perp = k+1$ y por tanto $\mathcal{C}^\perp$ es un código MDS. $\square$

Capítulo 2

Códigos Reed-Solomon, Alternantes y Goppa

2.1. Códigos Reed-Solomon

Los códigos Reed-Solomon se pueden describir como códigos BCH sobre $\mathbb{F}_q$ de longitud $n = q-1$. De esta forma se desarrolla en [4]. Sin embargo, los códigos Reed-Solomon también se pueden definir sin necesidad de introducir los códigos BCH. Es así como se desarrolla en [2] y como se procederá en el trabajo.

La importancia de los códigos Reed-Solomon reside en que son códigos MDS, por lo que tienen la mayor distancia mínima posible, en virtud de la cota de Singleton. Además, son convenientes para construir otros códigos, como por ejemplo los códigos Goppa que veremos más adelante. También, son útiles para corregir errores ya que admiten algoritmos de descodificación con complejidad polinómica. Sin embargo, la desventaja que tienen los códigos Reed-Solomon es que requieren tamaños de cuerpo mayores que la longitud del código, es decir, $n \leq q$, y esto supone un problema pues si n es grande entonces q también lo es, y por lo tanto estos códigos requieren operaciones sobre cuerpos finitos grandes, que son operaciones menos eficientes.

2.1.1. Definición y parámetros fundamentales

A lo largo de toda la sección se denotará por $\mathbb{P}_{k,q}$ al conjunto de todos los polinomios de $\mathbb{F}_q[x]$ de grado menor que k, es decir, $\mathbb{P}_{k,q} = \{f \in \mathbb{F}_q[x] : \deg(f) < k\}$. Nótese que $\mathbb{P}_{k,q}$ es un $\mathbb{F}_q$-espacio vectorial de dimensión k.

Definición 2.1.1.1. Sea $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_q^n$ con $\alpha_i \neq \alpha_j$ para todo $i \neq j$ (luego $n \leq q$). Dado $k \leq n$, consideramos $\mathbb{P}_{k,q}$. Un código Reed-Solomon es un código de la forma:

$$RS_{k,n}(\boldsymbol{\alpha}) = \{(f(\alpha_1), \ldots, f(\alpha_n)) : f \in \mathbb{P}_{k,q}\} \subset \mathbb{F}_q^n.$$

Lema 2.1.1.1. Los códigos Reed-Solomon son códigos lineales.

Demostración. Sean $\mathbf{c} = (f(\alpha_1), \ldots, f(\alpha_n))$, $\mathbf{c'} = (g(\alpha_1), \ldots, g(\alpha_n)) \in RS_{k,n}(\boldsymbol{\alpha})$.

Para cada $a, b \in \mathbb{F}_q$ se tiene que $a\mathbf{c} + b\mathbf{c'} = (h(\alpha_1), \ldots, h(\alpha_n)) \in RS_{k,n}(\boldsymbol{\alpha})$, donde $h(x) = af(x) + bg(x) \in \mathbb{P}_{k,q}$ por ser $f, g \in \mathbb{P}_{k,q}$ y $\mathbb{P}_{k,q}$ un $\mathbb{F}_q$-espacio vectorial.

Como hemos probado que $RS_{k,n}(\boldsymbol{\alpha})$ es un subespacio vectorial sobre $\mathbb{F}_q$, entonces se tiene que es un código lineal. □

Lema 2.1.1.2. $RS_{k,n}(\boldsymbol{\alpha})$ tiene longitud n y dimensión k.

Demostración. Por definición del código $RS_{k,n}(\boldsymbol{\alpha})$ es claro que su longitud es n.

Veamos que tiene dimensión k. Consideramos la aplicación $ev : \mathbb{P}_{k,q} \longrightarrow \mathbb{F}_q^n$ dada por $ev(f) = (f(\alpha_1), \ldots, f(\alpha_n))$, que es lineal entre dos espacios vectoriales sobre $\mathbb{F}_q$. Además, $\mathrm{Im}(ev) = RS_{k,n}(\boldsymbol{\alpha})$.

Veamos que ev es inyectiva probando que $\ker(ev) = \{0\}$:

Sea $f \in \ker(ev)$, entonces $ev(f) = (0, \ldots, 0)$, es decir, $f(\alpha_1) = \ldots = f(\alpha_n) = 0$, por lo que f tiene n raíces distintas. Pero $\deg(f) < k \leq n$, luego debe ser $f \equiv 0$.

Se tiene por lo tanto un isomorfismo $ev : \mathbb{P}_{k,q} \longrightarrow RS_{k,n}(\boldsymbol{\alpha})$ de espacios vectoriales sobre $\mathbb{F}_q$ y dado que $\dim(\mathbb{P}_{k,q}) = k$, se deduce que $\dim(RS_{k,n}(\boldsymbol{\alpha})) = k$. □

Teorema 2.1.1.1. $RS_{k,n}(\boldsymbol{\alpha})$ es un código MDS. Por lo tanto, $RS_{k,n}(\boldsymbol{\alpha})$ es un código lineal sobre $\mathbb{F}_q$ de tipo $[n, k, n-k+1]$.

Demostración. Sea $\mathbf{c} = (f(\alpha_1), \ldots, f(\alpha_n)) \in RS_{k,n}(\boldsymbol{\alpha})$.

Si $\mathbf{c}$ tiene una coordenada nula, por ejemplo c_i, entonces α_i es una raíz del polinomio $f \in \mathbb{P}_{k,q}$. Y como $\deg(f) < k$, $\mathbf{c}$ puede tener a lo más $k-1$ coordenadas nulas, es decir, que $w(\mathbf{c}) \geq n - (k-1)$. Esta desigualdad se cumple en particular para el mínimo de los pesos de una palabra del código, por lo que $d \geq n - k + 1$. Además, por la cota de Singleton, $d \leq n - k + 1$.

Se concluye así que $RS_{k,n}(\boldsymbol{\alpha})$ es un código MDS. □

2.1.2. Codificación de los Códigos Reed-Solomon

Sean $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ los elementos fijos que describen el código $RS_{k,n}(\boldsymbol{\alpha})$. Supongamos que queremos transmitir el mensaje $(u_0, u_1, \ldots, u_{k-1}) \in \mathbb{F}_q^k$. Esta información la podemos interpretar como un polinomio de grado menor que k, es decir, $u(x) = u_0 + u_1 x + \ldots + u_{k-1}x^{k-1} \in \mathbb{P}_{k,q}$. Entonces, para codificar los k dígitos calculamos $(u(\alpha_1), u(\alpha_2), \ldots, u(\alpha_n)) \in RS_{k,n}(\boldsymbol{\alpha})$.

Fijada una base de $\mathbb{P}_{k,q}$, por ejemplo $\{1, x, \ldots, x^{k-1}\}$, la matriz generadora de

un código $RS_{k,n}(\boldsymbol{\alpha})$ es de la forma

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}. \tag{2.1}$$

Nótese que al multiplicar $(u_0, u_1, \ldots, u_{k-1})$ por G, se hacen un total de kn multiplicaciones, por lo que este proceso de codificación tiene complejidad $O(kn)$, es decir, cuadrática.

Nota. Se denotará por $X_{k,n}$ a la matriz de la forma (2.1), a lo largo de todo el trabajo. Obsérvese que si $k = n$, entonces $X_{n,n}$ es una matriz de tipo Vandermonde.

Se desarrollarán a continuación los códigos Reed-Solomon generalizados. Estos códigos tienen la particularidad de que su dual también es un código Reed-Solomon generalizado, lo cual no se cumple en general para códigos Reed-Solomon (no generalizados). Además, como los códigos Reed-Solomon son un caso particular de los Reed-Solomon generalizados, para obtener una matriz de control de $RS_{k,n}(\boldsymbol{\alpha})$, se calculará su dual y se hallará una matriz generadora del dual.

2.1.3. Códigos Reed-Solomon Generalizados

Los códigos Reed-Solomon generalizados los necesitaremos para poder estudiar después los códigos Goppa, los cuales utilizó McEliece para describir su criptosistema.

Definición 2.1.3.1. Sean $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$ con $\alpha_i \neq \alpha_j$, si $i \neq j$ y $\mathbf{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_{q^m}^n$ con $v_i \neq 0$, $1 \leq i \leq n$.

Se dice que $\mathcal{C}$ es un código Reed-Solomon generalizado si

$$\mathcal{C} = GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v}) = \{(v_1 f(\alpha_1), v_2 f(\alpha_2), \ldots, v_n f(\alpha_n)) : f \in \mathbb{P}_{k,q^m}\}.$$

Nótese que si $v_i = 1$, para todo $i = 1, 2, \ldots, n$, entonces se tiene un código Reed-Solomon.

Proposición 2.1.3.1. Los códigos $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ son códigos sobre $\mathbb{F}_{q^m}$ de tipo $[n, k, n-k+1]$, por lo que también son códigos MDS.

Demostración. $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ y $RS_{k,n}(\boldsymbol{\alpha})$ son códigos equivalentes con matriz monomial[1] $\mathrm{diag}(\mathbf{v})$, y por lo tanto, tienen los mismos parámetros fundamentales.

Dado que $RS_{k,n}(\boldsymbol{\alpha})$ es de tipo $[n, k, n-k+1]$, $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ también. □

[1]Ver definición 2.2.1.3, subsección 2.2.1. Matriz Generadora.

Teorema 2.1.3.1. El dual de $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ es $GRS_{n-k,n}(\boldsymbol{\alpha}, \mathbf{y})$ para algún $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_{q^m}^n$ con $y_i \neq 0$, $1 \leq i \leq n$.

Demostración. Supongamos que $k = n - 1$ y sea $\mathcal{D} = GRS_{n-1,n}(\boldsymbol{\alpha}, \mathbf{v})^{\perp}$. Entonces $\dim \mathcal{D} = n - (n-1) = 1$, por lo que $\mathcal{D}$ es el conjunto de los múltiplos de algún vector fijo $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_{q^m}^n$. Si $y_i \neq 0$ para todo $i = 1, \ldots, n$, entonces $\mathcal{D} = GRS_{1,n}(\boldsymbol{\alpha}, \mathbf{y})$, y se deduce que para $k = n - 1$, $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})^{\perp} = GRS_{n-k,n}(\boldsymbol{\alpha}, \mathbf{y})$.

Sabemos que los códigos Reed-Solomon generalizados son códigos MDS, entonces por la proposición 1.2.5.1, como el código dual de un código MDS es también un código MDS, se tiene que $\mathcal{D}$ es un código MDS, por lo que $d(\mathcal{D}) = n - 1 + 1 = n$. Como $\mathbf{y} \in \mathcal{D}$, se tiene que $n = d(\mathcal{D}) \leq w(\mathbf{y}) \leq n$, de donde se sigue que $w(\mathbf{y}) = n$, por lo que $y_i \neq 0$ para todo $i = 1, \ldots, n$.

Consideremos una matriz generadora de $GRS_{n-1,n}(\boldsymbol{\alpha}, \mathbf{v})$, $G = X_{n-1,n}\mathrm{diag}(\mathbf{v})$. Entonces G es una matriz de control para $GRS_{1,n}(\boldsymbol{\alpha}, \mathbf{y})$, y como $\mathbf{y} \in GRS_{1,n}(\boldsymbol{\alpha}, \mathbf{y})$, se satisface que $G\mathbf{y}^t = \mathbf{0}^t$, es decir,

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-2} & \alpha_2^{n-2} & \ldots & \alpha_n^{n-2} \end{pmatrix} \begin{pmatrix} v_1 y_1 \\ v_2 y_2 \\ \vdots \\ v_n y_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{2.2}$$

Supongamos ahora que $k < n - 1$ y sean $s \leq k - 1$, $t \leq n - k - 1$, entonces

$$\sum_{i=1}^{n} (\alpha_i^s v_i)(\alpha_i^t y_i) = \sum_{i=1}^{n} \alpha_i^{s+t} v_i y_i = 0,$$

donde la última igualdad es consecuencia de que $s + t \leq n - 2$ y de (2.2).

Esto implica que $GRS_{n-k,n}(\boldsymbol{\alpha}, \mathbf{y}) \subset GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})^{\perp}$, y como ambos códigos tienen la misma dimensión, se deduce que son iguales.

□

Considerando $r = n - k$, unas matrices generadora y de control del código $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ son, respectivamente:

$$G = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \cdots & v_n\alpha_n \\ v_1\alpha_1^2 & v_2\alpha_2^2 & \cdots & v_n\alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \cdots & v_n\alpha_n^{k-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} v_1 & & & 0 \\ & v_2 & & \\ & & \ddots & \\ 0 & & & v_n \end{pmatrix},$$

$$H = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_1\alpha_1 & y_2\alpha_2 & \cdots & y_n\alpha_n \\ y_1\alpha_1^2 & y_2\alpha_2^2 & \cdots & y_n\alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ y_1\alpha_1^{r-1} & y_2\alpha_2^{r-1} & \cdots & y_n\alpha_n^{r-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} y_1 & & & 0 \\ & y_2 & & \\ & & \ddots & \\ 0 & & & y_n \end{pmatrix},$$

donde $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ con $y_i \in \mathbb{F}_{q^m}$ no nulo para cada $i = 1, 2, \ldots, n$, es tal que $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp = GRS_{n-k,n}(\boldsymbol{\alpha}, \mathbf{y})$.

Como $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp = GRS_{r,n}(\boldsymbol{\alpha}, \mathbf{y})$ y H es una matriz de control de $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$, se tiene que H es también una matriz generadora para $GRS_{r,n}(\boldsymbol{\alpha}, \mathbf{y})$.

2.1.4. Descodificación de los Códigos Reed-Solomon y Reed-Solomon generalizados

Primero se verán una serie de resultados y después se introducirá un algoritmo de descodificación única, que es específico para códigos Reed-Solomon. Por último, se adaptará tal algoritmo a los códigos Reed-Solomon generalizados.

Tenemos que $t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$ es el máximo número de errores que puede corregir un $RS_{k,n}(\boldsymbol{\alpha})$. Supongamos que $\mathbf{c} \in RS_{k,n}(\boldsymbol{\alpha})$, que $\mathbf{e} \in \mathbb{F}_q^n$ es el vector de error con $w(\mathbf{e}) \leq t$ y $\mathbf{r} = \mathbf{c} + \mathbf{e} = (r_1, \ldots, r_n)$ es la palabra recibida.

Teorema 2.1.4.1. Existe un polinomio no nulo

$$Q(x, y) = Q_0(x) + yQ_1(x) \in \mathbb{F}_q[x, y],$$

tal que:

- $Q(\alpha_i, r_i) = 0$ para todo $i = 1, 2, \ldots, n$,
- $\deg(Q_0) \leq n - 1 - t$,
- $\deg(Q_1) \leq n - 1 - t - (k - 1)$.

Demostración. Se buscan

$$Q_0(x) = \sum_{j=0}^{n-1-t} q_{0j}x^j, \qquad Q_1(x) = \sum_{k=0}^{n-1-t-(k-1)} q_{1k}x^k,$$

tales que $Q(\alpha_i, r_i) = Q_0(\alpha_i) + r_iQ_1(\alpha_i) = 0$ para todo $i = 1, 2, \ldots, n$.

Se tiene así un sistema lineal homogéneo de n ecuaciones, que sabemos que tiene solución no nula si el número de incógnitas es mayor que n, el número de ecuaciones.

Como $\deg(Q_0) \leq n-1-t$ y $\deg(Q_1) \leq n-1-t-(k-1)$ se tienen $(n-1-t)+1+(n-1-t-(k-1))+1 = 2n-2t-(k-1)$ incógnitas.

Dado que $t = \lfloor \frac{n-k}{2} \rfloor$, se sigue que $2n-2t-(k-1) \geq 2n-(n-k)-(k-1) = n+1 > n$. Luego se tienen más incógnitas que ecuaciones y por lo tanto, existe solución nula, es decir, se tiene que existen tales polinomios Q_0 y Q_1 no nulos. □

Definición 2.1.4.1. El polinomio $Q(x,y) = Q_0(x) + yQ_1(x)$ del teorema anterior se denomina polinomio interpolador de la palabra recibida.

Teorema 2.1.4.2. Si $w(\mathbf{e}) \leq t = \lfloor \frac{d-1}{2} \rfloor$ y $\mathbf{c}$ fue generada por el polinomio $f \in \mathbb{P}_{k,q}$, entonces se tiene que $f(x) = -\frac{Q_0(x)}{Q_1(x)}$.

Demostración. Por hipótesis $\mathbf{c} = (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))$, luego si $\mathbf{e} = (e_1, e_2, \ldots, e_n)$, se tiene que $\mathbf{r} = (r_1, r_2, \ldots, r_n)$, con $r_i = f(\alpha_i) + e_i$, $i = 1, 2, \ldots, n$.

Por el teorema anterior, sabemos que siempre existe el polinomio interpolador $Q(x,y) = Q_0(x)+yQ_1(x)$. Entonces, para todo $i = 1, 2, \ldots, n$ se tiene que $Q(\alpha_i, r_i) = Q(\alpha_i, f(\alpha_i)+e_i) = 0$. Como $w(\mathbf{e}) \leq t$, se tiene que $e_i = 0$ para al menos $n-t$ valores de i. Entonces, el polinomio $Q(x, f(x)) = Q_0(x) + f(x)Q_1(x) \in \mathbb{F}_q[x]$ se anula al menos en $n-t$ de los α_i, es decir, que $Q(x, f(x))$ tiene al menos $n-t$ raíces.

Por otro lado, como $\deg(Q_0(x)) \leq n-t-1$, $\deg(Q_1(x)) \leq n-t-1-(k-1)$ y $\deg(f(x)) \leq k-1$, se deduce que $\deg(Q(x, f(x))) \leq n-t-1$. Por lo tanto, ha de ser $Q(x, f(x)) = Q_0(x) + f(x)Q_1(x) = 0$.

Si vemos que $Q_1(x)$ es no nulo entonces $f(x) = -\frac{Q_0(x)}{Q_1(x)}$ como se quería.

Si fuese $Q_1 \equiv 0$, entonces $Q(x,y) = Q_0(x)$. Como $0 = Q(\alpha_i, r_i) = Q_0(\alpha_i)$, para todo $i = 1, 2, \ldots, n$, entonces Q_0 tiene n raíces, pero $\deg(Q_0) \leq n-t-1 < n$. Por lo tanto, solo puede ser $Q_0 \equiv 0$, de donde se sigue que $Q \equiv 0$, lo que es absurdo pues el polinomio interpolador es no nulo. □

Nótese que $Q(x,y) = Q_0(x) + yQ_1(x) = Q_1(x)(y + \frac{Q_0(x)}{Q_1(x)}) = Q_1(x)(y - f(x))$. Además, para cada $i = 1, 2, \ldots, n$ se tiene que $0 = Q(\alpha_i, r_i) = Q_1(\alpha_i)(r_i - f(\alpha_i)) = Q_1(\alpha_i)e_i$. Entonces, si $e_i \neq 0$, se deduce que α_i es una raíz de $Q_1(x)$.

En conclusión, las raíces α_i de $Q_1(x)$, permiten determinar los índices i donde se ha producido un error en la transmisión.

Definición 2.1.4.2. El polinomio $Q_1(x)$ se denomina polinomio detector de errores.

El algoritmo 2 RSDecoder que describimos a continuación, es un algoritmo de descodificación única para los códigos Reed-Solomon.

Algorithm 2: RSDecoder

Input : El código $RS_{k,n}(\boldsymbol{\alpha})$ t-corrector con $l_0 = n-1-t$ y $l_1 = n-1-t-(k-1)$.
La palabra recibida $\mathbf{r} = (r_1, r_2, \ldots, r_n)$.

Output: La palabra descodificada $\mathbf{c} \in RS_{k,n}(\boldsymbol{\alpha})$, o **error**.

1 Encontrar una solución no nula del sistema lineal

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{l_0} & r_1 & r_1\alpha_1 & \cdots & r_1\alpha_1^{l_1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{l_0} & r_2 & r_2\alpha & \cdots & r_2\alpha_2^{l_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{l_0} & r_n & r_n\alpha_n & \cdots & r_n\alpha_n^{l_1} \end{pmatrix} \begin{pmatrix} q_{00} \\ q_{01} \\ q_{02} \\ \vdots \\ q_{0l_0} \\ q_{10} \\ q_{11} \\ \vdots \\ q_{1l_1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

2 $Q_0(x) := \sum_{j=0}^{l_0} q_{0j}x^j$

3 $Q_1(x) := \sum_{j=0}^{l_1} q_{1j}x^j$

4 $g(x) := -\frac{Q_0(x)}{Q_1(x)}$

5 **if** $g(x) \in \mathbb{P}_{k,q}$ **and** $d(\boldsymbol{r}, (g(\alpha_1), g(\alpha_2), \ldots, g(\alpha_n))) \leq t$ **then**

6 **return** $\mathbf{c} = (g(\alpha_1), g(\alpha_2), \ldots, g(\alpha_n))$

7 **else**

8 **return error**

Proposición 2.1.4.1. Si se han producido como mucho t errores durante la transmisión, entonces, el output del algoritmo 2 RSDecoder es la palabra enviada.

Demostración. Es consecuencia del teorema 2.1.4.2. □

En el algoritmo 2 RSDecoder se resuelve un sistema de ecuaciones lineales y se hace una división de polinomios. Estas operaciones tienen complejidades cúbica y cuadrática respectivamente. Por lo tanto, el algoritmo 2 RSDecoder tiene complejidad computacional cúbica, $O(n^3)$.

Consideremos una matriz generadora de $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$, $G = X_{k,n} \cdot \text{diag}(\mathbf{v})$. Como $X_{k,n}$ es una matriz generadora de $RS_{k,n}(\boldsymbol{\alpha})$, se tiene que si $\mathbf{c} \in GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$, entonces $\mathbf{c}' = \mathbf{c} \cdot \text{diag}(v_1^{-1}, \ldots, v_n^{-1}) \in RS_{k,n}(\boldsymbol{\alpha})$. Nótese que tiene sentido considerar v_i^{-1}, pues para cada $i = 1, \ldots, n$, se tiene que $v_i \neq 0$ en el cuerpo $\mathbb{F}_{q^m}$.

A continuación, se adapta el algoritmo 2 RSDecoder para descodificar los códigos Reed-Solomon generalizados. La idea del algoritmo 3 GRSDecoder radica en que si $\mathbf{r} = \mathbf{c} + \mathbf{e}$, con $\mathbf{c} \in GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$, entonces

$$\mathbf{r} \cdot \text{diag}(\mathbf{v}^{-1}) = \mathbf{c} \cdot \text{diag}(\mathbf{v}^{-1}) + \mathbf{e} \cdot \text{diag}(\mathbf{v}^{-1}) = \mathbf{c}' + \mathbf{e}',$$

y como $\mathbf{c}' \in RS_{k,n}(\boldsymbol{\alpha})$, y $w(\mathbf{e}') = w(\mathbf{e})$, se puede utilizar el algoritmo 2 RSDecoder cuyo output nos proporciona **error** o $\mathbf{c}'$, en cuyo caso, solo resta multiplicar por $\text{diag}(\mathbf{v})$ para obtener $\mathbf{c}$.

Algorithm 3: GRSDecoder

Input : El código $GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$ t-corrector y la palabra recibida $\mathbf{r} = (r_1, r_2, \ldots, r_n)$.

Output: La palabra descodificada $\mathbf{c} \in GRS_{k,n}(\boldsymbol{\alpha}, \mathbf{v})$, o **error**.

$\mathbf{r}' := (r_1 v_1^{-1}, r_2 v_2^{-1}, \ldots, r_n v_n^{-1})$
if RSDecoder$(RS_{k,n}(\boldsymbol{\alpha}), t, \mathbf{r}')$==**error then**
  return error
else
  return $\mathbf{c}$ = RSDecoder$(RS_{k,n}(\boldsymbol{\alpha}), t, \mathbf{r}') \cdot \text{diag}(\mathbf{v})$

En el algoritmo 3 GRSDecoder, RSDecoder$(RS_{k,n}(\boldsymbol{\alpha}), t, \mathbf{r}')$ se refiere a que se aplica el algoritmo 2 RSDecoder a la palabra $\mathbf{r}'$ con el código $RS_{k,n}(\boldsymbol{\alpha})$ t-corrector.

Nótese que el algoritmo 3 GRSDecoder tiene complejidad cúbica, $O(n^3)$, ya que el multiplicar por $v_1, v_2, \ldots, v_n$ tiene complejidad $O(n)$ y la complejidad computacional del algoritmo 2 RSDecoder es $O(n^3)$.

2.2. Códigos Alternantes

A lo largo de esta sección se dará una pequeña introducción sobre los códigos alternantes. Veremos un algoritmo de descodificación que podremos adaptar después a los códigos Goppa, por ser estos últimos un caso particular de códigos alternantes. Para ver un tratado más desarrollado de los códigos alternantes, referimos al lector a [4].

Los códigos alternantes están estrechamente relacionados con los códigos Reed-Solomon generalizados

$$GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) = \{(v_1 f(\alpha_1), v_2 f(\alpha_2), \ldots, v_n f(\alpha_n)) : f \in \mathbb{P}_{k_0,q^m}\}.$$

Definición 2.2.1. Sean $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$ con $\alpha_i \neq \alpha_j$, si $i \neq j$ e $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ con $y_i \in \mathbb{F}_{q^m}$ no nulo para cada $i = 1, 2, \ldots, n$.

Se denomina código alternante a un código de la forma

$$\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = \{\mathbf{c} = (c_1, c_2, \ldots, c_n) \in GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) : c_i \in \mathbb{F}_q, \quad i = 1, \ldots, n\},$$

es decir,

$$\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n.$$

El parámetro $\mathbf{y}$ que define el código alternante, es tal que $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp = GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})$. Por lo tanto, $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) = (GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp)^\perp = GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp$, y se tiene que

$$\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = \{(c_1, c_2, \ldots, c_n) \in GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp : c_i \in \mathbb{F}_q, \quad i = 1, \ldots, n\}.$$

Sea $H \in \mathfrak{M}_{(n-k_0)\times n}(\mathbb{F}_{q^m})$ una matriz de control de $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$. Una definición equivalente de código alternante es $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = \{\mathbf{a} \in \mathbb{F}_q^n : H\mathbf{a}^t = \mathbf{0}^t\}$, de donde se deduce que H es también matriz de control para $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ sobre $\mathbb{F}_{q^m}$.

Por ejemplo, una matriz de control para $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ sobre $\mathbb{F}_{q^m}$ es

$$H = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_1\alpha_1 & y_2\alpha_2 & \cdots & y_n\alpha_n \\ y_1\alpha_1^2 & y_2\alpha_2^2 & \cdots & y_n\alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ y_1\alpha_1^{r-1} & y_2\alpha_2^{r-1} & \cdots & y_n\alpha_n^{r-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} y_1 & & & 0 \\ & y_2 & & \\ & & \ddots & \\ 0 & & & y_n \end{pmatrix}. \tag{2.3}$$

Lema 2.2.1. Sea $\mathcal{C}$ un código cualquiera sobre $\mathbb{F}_q$. Si $H \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_{q^m})$ es una matriz de control de $\mathcal{C}$, entonces existe una matriz de control de $\mathcal{C}$, $\bar{H} \in \mathfrak{M}_{((n-k)m)\times n}(\mathbb{F}_q)$.

Demostración. Consideremos una matriz de control de $\mathcal{C}$,

$$H = \begin{pmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k} \end{pmatrix} \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_{q^m}),$$

donde $\mathbf{h}_i$ denota la fila i-ésima de H, $1 \leq i \leq n-k$.

Sea $\{\beta_1, \ldots, \beta_m\}$ una base de $\mathbb{F}_{q^m}$ sobre $\mathbb{F}_q$. Entonces, $\mathbf{h}_i = \sum_{j=1}^{m} \beta_j \mathbf{h}_{ij}$, donde $\mathbf{h}_{ij} \in \mathbb{F}_q^n$ para todo $i \in \{1, \ldots, n-k\}$ y todo $j \in \{1 \ldots, m\}$.

Por ser H matriz de control de $\mathcal{C}$, se tiene que $\mathbf{c} \in \mathcal{C}$ si y solo si $\mathbf{c}H^t = \mathbf{0}$, es decir, $0 = \mathbf{ch}_i^t = \sum_{j=1}^{m} \beta_j(\mathbf{ch}_{ij}^t)$ para cada $i = 1, \ldots, n-k$, lo cual equivale a que $\mathbf{ch}_{ij}^t = 0$ para todo $i = 1, \ldots, n-k$, y todo $j = 1, \ldots, m$, pues $\mathbf{ch}_{ij}^t \in \mathbb{F}_q$ y los β_j son $\mathbb{F}_q$-linealmente independientes.

Lo anterior es equivalente a que $\mathbf{c}\bar{H}^t = \mathbf{0}$, donde

$$\bar{H} = \begin{pmatrix} \mathbf{h}_{11} \\ \vdots \\ \mathbf{h}_{1m} \\ \vdots \\ \mathbf{h}_{n-k,1} \\ \vdots \\ \mathbf{h}_{n-k,m} \end{pmatrix},$$

es una matriz con $(n-k)m$ filas, n columnas y sus coeficientes son elementos de $\mathbb{F}_q$, es decir, $\bar{H}$ es una matriz de control de $\mathcal{C}$. □

Proposición 2.2.1. Los códigos alternantes, $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n$, son códigos lineales sobre $\mathbb{F}_q$ de tipo $[n, k, d]$ tales que:

i) $n - mr \leq k \leq n - r$,

ii) $d \geq r + 1$,

con $r = n - k_0$.

Demostración. $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ son códigos lineales sobre $\mathbb{F}_q$ con longitud n, por ser la restricción de $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$ a $\mathbb{F}_q$.

Veamos que $n - mr \leq k \leq n - r$:

Dado que $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n$, su dimensión será como mucho la del cógido Reed-Solomon generalizado, es decir, $k \leq k_0 = n - (n - k_0) = n - r$.

Veamos la otra desigualdad:

Sea $H \in \mathfrak{M}_{(n-k_0)\times n}(\mathbb{F}_{q^m})$ una matriz de control de $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$. Entonces, H también es una matriz de control para $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$. Por el lema 2.2.1, existe $\bar{H} \in \mathfrak{M}_{((n-k_0)m)\times n}(F_q)$ matriz de control de $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$.

Por ser $\bar{H}$ matriz de control de $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$, pero no necesariamente de rango máximo, tiene $(n - k_0)m = rm$ filas. Y como el rango de $\bar{H}$ debe ser $n - k$, se deduce que $n - k \leq (n - k_0)m = rm$, es decir, $n - rm \leq k$.

Veamos por último que $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ tiene distancia mínima $d \geq r + 1$:

Como $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n \subset GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$, se tiene que el peso mínimo de $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ será como mínimo el peso mínimo de $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$. Además, como los códigos Reed-Solomon generalizados son códigos MDS, se deduce que $d = d(\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})) \geq n - k_0 + 1 = r + 1$.

□

2.2.1. Código dual de un código alternante

Para poder desarrollar el código dual de un código alternante, necesitamos introducir previamente algunos conceptos y resultados sobre los códigos traza.

Definición 2.2.1.1. La aplicación $T_{m,q} : \mathbb{F}_{q^m} \longrightarrow \mathbb{F}_q$ dada por

$$T_{m,q}(x) = x + x^q + \ldots + x^{q^{m-1}}, \qquad x \in \mathbb{F}_{q^m},$$

se denomina aplicación traza.

Si no surge ambigüedad sobre q, denotaremos por T_m a la aplicación traza.

Proposición 2.2.1.1. La aplicación traza T_m es una aplicación $\mathbb{F}_q$-lineal.

Demostración. Sean $x, y \in \mathbb{F}_{q^m}$.

$$\begin{aligned} T_m(x) + T_m(y) &= (x + x^q + \ldots + x^{q^{m-1}}) + (y + y^q + \ldots + y^{q^{m-1}}) \\ &= (x + y) + (x^q + y^q) + \ldots + (x^{q^{m-1}} + y^{q^{m-1}}) \\ &= (x + y) + (x + y)^q + \ldots + (x + y)^{q^{m-1}} \\ &= T_m(x + y). \end{aligned}$$

De donde la penúltima igualdad es consecuencia del binomio de Newton aplicado sobre elementos de un cuerpo finito $\mathbb{F}_{q^m}$, con $q = p^s$ para algún primo p y un entero positivo s.

Sean $x \in \mathbb{F}_{q^m}$, $\lambda \in \mathbb{F}_q = \{z : z^q = z\}$. Entonces, se tiene que $\lambda^{q^j} = (\lambda^q)^{q^{j-1}} = \lambda^{q^{j-1}} = \ldots = \lambda$, para todo $j \in \mathbb{Z}_+$. Por lo que

$$\begin{aligned} \lambda T_m(x) &= \lambda(x + x^q + \ldots + x^{q^{m-1}}) \\ &= \lambda x + \lambda x^q + \ldots + \lambda x^{q^{m-1}} \\ &= (\lambda x) + (\lambda x)^q + \ldots + (\lambda x)^{q^{m-1}} \\ &= T_m(\lambda x). \end{aligned}$$

Así queda probado que T_m es una aplicación $\mathbb{F}_q$-lineal. □

Definición 2.2.1.2. Sea $\mathcal{C}$ un código de tipo $[n, k^*, d^*]$ sobre $\mathbb{F}_{q^m}$. Se define el código traza sobre $\mathbb{F}_q$ como

$$T_m(\mathcal{C}) = \{T_m(\mathbf{b}) = (T_m(b_1), \ldots, T_m(b_n)) : \mathbf{b} = (b_1, \ldots, b_n) \in \mathcal{C}\}.$$

$T_m(\mathcal{C})$ es un código de tipo $[n, k, d]$ sobre $\mathbb{F}_q$. Se puede demostrar que $k^* \leq k \leq mk^*$ y $d^* \leq d$, pero debido a que no haremos uso de estas desigualdades no daremos su demostración, la cual se puede encontrar en [4].

La aplicación traza T_m se puede usar para expresar el dual de $\mathcal{C} \cap \mathbb{F}_q^n$ en términos de $\mathcal{C}^\perp$ como muestra el teorema de Delsarte.

Teorema 2.2.1.1. (Delsarte)
El dual de $\mathcal{C} \cap \mathbb{F}_q^n$ es el código traza de $\mathcal{C}^\perp$. Es decir, $(\mathcal{C} \cap \mathbb{F}_q^n)^\perp = T_m(\mathcal{C}^\perp)$.

Demostración. Sea $\mathbf{a} \in T_m(\mathcal{C}^\perp)$, entonces existe $\mathbf{b} = (b_1, \ldots, b_n) \in \mathcal{C}^\perp$ tal que $\mathbf{a} = (T_m(b_1), \ldots, T_m(b_n))$.

Tenemos que $\mathbf{a} \in (\mathcal{C} \cap \mathbb{F}_q^n)^\perp$ si y solo si $H\mathbf{a}^t = \mathbf{0}^t$, donde H es una matriz de control de $(\mathcal{C} \cap \mathbb{F}_q^n)^\perp$, es decir, una matriz generadora de $\mathcal{C} \cap \mathbb{F}_q^n$. Luego $\mathbf{a} \in (\mathcal{C} \cap \mathbb{F}_q^n)^\perp$ si y solo si para todo $\mathbf{c} \in \mathcal{C} \cap \mathbb{F}_q^n$ se tiene que $\mathbf{c} \cdot \mathbf{a}^t = 0$.

Sea $\mathbf{c} \in \mathcal{C} \cap \mathbb{F}_q^n$,

$$\mathbf{c} \cdot \mathbf{a}^t = \sum_{i=1}^{n} c_i T_m(b_i) = T_m\left(\sum_{i=1}^{n} c_i b_i\right) = T_m(0) = 0,$$

donde la segunda igualdad se da por ser $\mathbf{c} \in \mathbb{F}_q^n$ y T_m una aplicación $\mathbb{F}_q$-lineal; la tercera igualdad se da por ser $\mathbf{c} \in \mathcal{C}$ y $\mathbf{b} \in \mathcal{C}^\perp$; y la última igualdad por ser T_m una aplicación $\mathbb{F}_q$-lineal.

Por lo tanto, se concluye que $T_m(\mathcal{C}^\perp) \subset (\mathcal{C} \cap \mathbb{F}_q^n)^\perp$.

Veamos que se tiene la otra contención, que es equivalente a que $(T_m(\mathcal{C}^\perp))^\perp \subset \mathcal{C} \cap \mathbb{F}_q^n$.

Sea $\mathbf{a} \in (T_m(\mathcal{C}^\perp))^\perp$. Como el código traza es un código sobre $\mathbb{F}_q$, también lo será su dual, por lo que $\mathbf{a} \in (T_m(\mathcal{C}^\perp))^\perp \subset \mathbb{F}_q^n$, y solo queda probar que $\mathbf{a} \in \mathcal{C}$, lo cual se cumple si $\mathbf{a} \cdot \mathbf{b} = 0$, para todo $\mathbf{b} \in \mathcal{C}^\perp$.

Sea $\mathbf{b} = (b_1, \ldots, b_n) \in \mathcal{C}^\perp$, entonces, $\lambda\mathbf{b} \in \mathcal{C}^\perp$ para todo $\lambda \in \mathbb{F}_{q^m}$, luego $T_m(\lambda\mathbf{b}) \in T_m(\mathcal{C}^\perp)$ y como $\mathbf{a} \in (T_m(\mathcal{C}^\perp))^\perp$, se tiene que

$$0 = \mathbf{a} \cdot T_m(\lambda\mathbf{b}) = \sum_{i=1}^{n} a_i T_m(\lambda b_i) = T_m\left(\lambda \sum_{i=1}^{n} a_i b_i\right), \quad \forall \lambda \in \mathbb{F}_{q^m}.$$

Si $x = \sum_{i=1}^{n} a_i b_i \neq 0$, entonces λx es raíz de T_m para todo $\lambda \in \mathbb{F}_{q^m}$, pero $\{\lambda x : \lambda \in \mathbb{F}_{q^m}\} = \mathbb{F}_{q^m}$, por lo que todo elemento de $\mathbb{F}_{q^m}$ sería raíz de T_m, lo cual es absurdo

pues $\deg(T_m(y)) = q^{m-1} < q^m = |\mathbb{F}_{q^m}|$, viendo $T_m(y)$ como polinomio en y, por lo que T_m tiene como mucho q^{m-1} raíces en $\mathbb{F}_{q^m}$. Luego ha de ser $\sum_{i=1}^{n} a_i b_i = 0$ y se concluye así que $(T_m(\mathcal{C}^\perp))^\perp \subset \mathcal{C} \cap \mathbb{F}_q^n$.

□

A continuación, se describe el código dual de $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ en términos de un código traza.

Teorema 2.2.1.2. El dual de un código alternante $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ es

$$T_m(GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})) = T_m(GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp).$$

Demostración. Como $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n$, aplicando el teorema de Delsarte 2.2.1.1 y teniendo en cuenta que $GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp = GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})$, se tiene que

$$\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})^\perp = (GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n)^\perp = T_m(GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})^\perp) = T_m(GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})).$$

□

2.2.2. Descodificación de un código alternante

Vamos a ver que si tenemos un código $\mathcal{D} = \mathcal{C} \cap \mathbb{F}_q^n$, donde $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ es un código para el que tenemos un algoritmo de descodificación, entonces tal algoritmo nos servirá para descodificar $\mathcal{D}$. Después particularizaremos este caso para los códigos alternantes $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp \cap \mathbb{F}_q^n$.

Consideremos el código dado por $\mathcal{D} = \mathcal{C} \cap \mathbb{F}_q^n$, donde $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ es otro código. Sean $\mathbf{c} \in \mathcal{D} \subset \mathcal{C}$ la palabra enviada y $\mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n$ la palabra recibida. Si se han producido menos de $t = \lfloor \frac{d(\mathcal{C})-1}{2} \rfloor$ errores, entonces podemos aplicar el algoritmo de descodificación de $\mathcal{C}$ a $\mathbf{r}$, obteniendo así $\mathbf{c} \in \mathcal{C}$. Como $\mathcal{C} \subset \mathbb{F}_{q^m}^n$, también se puede ver $\mathbf{c}$ como un elemento de $\mathbb{F}_q^n$, por lo que $\mathbf{c} \in \mathcal{D}$.

Nota. Como $\mathcal{D} = \mathcal{C} \cap \mathbb{F}_q^n$, se tiene que $d(\mathcal{D}) \geq d(\mathcal{C})$, y el método descrito corrige hasta $\lfloor \frac{d(\mathcal{C})-1}{2} \rfloor$ errores en vez de $\lfloor \frac{d(\mathcal{D})-1}{2} \rfloor$, por lo que otro algoritmo de descodificación para $\mathcal{D}$ podría corregir más errores.

A continuación se dará un algoritmo de descodificación para códigos alternantes, el algoritmo 4 AlternantDecoder, cuyo input será la palabra recibida y el código t-corrector $GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp$, ya que $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp \cap \mathbb{F}_q^n$, y $GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^\perp = GRS_{k_0,n}(\boldsymbol{\alpha}, \mathbf{v})$, para algún $\mathbf{v} \in \mathbb{F}_q^n$ con $v_i \neq 0$, para todo $i = 1, \ldots, n$.

Algorithm 4: AlternantDecoder

Input : El código $GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y})^{\perp}$ t-corrector y la palabra recibida $\mathbf{r} \in \mathbb{F}_q^n$.

Output: La palabra descodificada $\mathbf{c} \in \mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$, o **error**.

```
if GRSDecoder(GRS_{n-k_0,n}(α, y)^⊥, t, r)==error then
    return error
else
    return c = GRSDecoder(GRS_{n-k_0,n}(α, y)^⊥, t, r)
```

El algoritmo 4 AlternantDecoder tiene complejidad cúbica, $O(n^3)$, por ser esta la complejidad computacional del algoritmo 3 GRSDecoder. Obsérvese que las operaciones realizadas en AlternantDecoder son en $\mathbb{F}_{q^m}$, no en $\mathbb{F}_q$.

2.3. Códigos Goppa

Los códigos Goppa son una de las subclases más interesantes de códigos alternantes. Además, son los únicos códigos que han resistido a todos los ataques contra el criptosistema de McEliece. Se describen en términos de un polinomio de Goppa $G(z)$. Primero daremos la definición en términos de polinomios de Goppa y después veremos que son códigos alternantes.

Definición 2.3.1. Sean $G(z)$ un polinomio de grado r con coeficientes en $\mathbb{F}_{q^m}$, y $L = \{\alpha_1, \ldots, \alpha_n\}$ un subconjunto de $\mathbb{F}_{q^m}$ tal que $G(\alpha_i) \neq 0$ para todo $\alpha_i \in L$.

Se dice que $\Gamma(L, G)$ es un código Goppa si

$$\Gamma(L, G) = \left\{ (a_1, \ldots, a_n) \in \mathbb{F}_q^n : \sum_{i=1}^{n} \frac{a_i}{z - \alpha_i} \equiv 0 \mod G(z) \right\}.$$

El polinomio $G(z)$ se denomina polinomio de Goppa; y si este es irreducible, entonces se dice que $\Gamma(L, G)$ es un código Goppa irreducible.

Nota. Normalmente, el subconjunto L es todo $\mathbb{F}_{q^m}$ salvo las raíces del polinomio de Goppa $G(z)$.

Observemos que $\Gamma(L, G)$ es un código lineal sobre $\mathbb{F}_q$ de longitud $|L| = n$. Entonces, a partir de su definición podemos encontrar una matriz de control.

Dado que $G(\alpha_i) \neq 0$, para todo $\alpha_i \in L$, se tiene que $z - \alpha_i$ no divide a $G(z)$, y por lo tanto $z - \alpha_i$ tiene inverso en $\mathbb{F}_{q^m}[z]/(G(z))$:

$$(z - \alpha_i)^{-1} = -\frac{G(z) - G(\alpha_i)}{z - \alpha_i} G(\alpha_i)^{-1},$$

ya que $-(z-\alpha_i)\dfrac{G(z)-G(\alpha_i)}{z-\alpha_i}G(\alpha_i)^{-1} \equiv -G(z)G(\alpha_i)^{-1}+1 \equiv 1 \mod G(z)$.

Así,

$$\mathbf{a} \in \Gamma(L,G) \quad \text{si y solo si} \quad \sum_{i=1}^{n} a_i \frac{G(z)-G(\alpha_i)}{z-\alpha_i} G(\alpha_i)^{-1} = 0, \tag{2.4}$$

como polinomio (no mód $G(z)$).

Si consideramos $G(z)=\sum_{i=0}^{r} g_i z^i$, con $g_i \in \mathbb{F}_{q^m}$ y $g_r \neq 0$, entonces

$$\frac{G(z)-G(\alpha_i)}{z-\alpha_i} = \frac{\sum_{j=0}^{r} g_j (z^j-\alpha_i^j)}{z-\alpha_i} = \sum_{j=1}^{r} g_j \left(\sum_{k=1}^{j} z^{j-k}\alpha_i^{k-1} \right) =$$

$$= g_r(z^{r-1}+z^{r-2}\alpha_i+\ldots+\alpha_i^{r-1})+g_{r-1}(z^{r-2}+z^{r-3}\alpha_i+\ldots+\alpha_i^{r-2})+\ldots+g_2(z-\alpha_i)+g_1.$$

Igualando los coeficientes de $z^{r-1}, z^{r-2}, \ldots, z, z^0$ a 0 en (2.4), se tiene que

$$\mathbf{a} \in \Gamma(L,G) \quad \text{si y solo si} \quad H\mathbf{a}^t = \mathbf{0}^t,$$

donde

$$H = \begin{pmatrix} g_r G(\alpha_1)^{-1} & g_r G(\alpha_2)^{-1} & \ldots & g_r G(\alpha_n)^{-1} \\ (g_{r-1}+\alpha_1 g_r)G(\alpha_1)^{-1} & (g_{r-1}+\alpha_2 g_r)G(\alpha_2)^{-1} & \ldots & (g_{r-1}+\alpha_n g_r)G(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ (\sum_{i=1}^{r} g_i \alpha_1^{i-1})G(\alpha_1)^{-1} & (\sum_{i=1}^{r} g_i \alpha_2^{i-1})G(\alpha_2)^{-1} & \ldots & (\sum_{i=1}^{r} g_i \alpha_n^{i-1})G(\alpha_n)^{-1} \end{pmatrix} =$$

$$= \begin{pmatrix} g_r & 0 & 0 & \ldots & 0 \\ g_{r-1} & g_r & 0 & \ldots & 0 \\ g_{r-2} & g_{r-1} & g_r & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ g_1 & g_2 & g_3 & \ldots & g_r \end{pmatrix} \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \ldots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} G(\alpha_1)^{-1} & & & 0 \\ & G(\alpha_2)^{-1} & & \\ & & \ddots & \\ 0 & & & G(\alpha_n)^{-1} \end{pmatrix} =$$

$$= CXY \in \mathfrak{M}_{r\times n}(\mathbb{F}_{q^m}).$$

Por definición de matriz de control, se concluye que una matriz de control del código Goppa $\Gamma(L,G)$ es la matriz H anterior.

Nótese que dado que C es una matriz invertible, otra matriz de control es

$$H' = XY = \begin{pmatrix} G(\alpha_1)^{-1} & G(\alpha_2)^{-1} & \ldots & G(\alpha_n)^{-1} \\ \alpha_1 G(\alpha_1)^{-1} & \alpha_2 G(\alpha_2)^{-1} & \ldots & \alpha_n G(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{r-1} G(\alpha_1)^{-1} & \alpha_2^{r-1} G(\alpha_2)^{-1} & \ldots & \alpha_n^{r-1} G(\alpha_n)^{-1} \end{pmatrix} \in \mathfrak{M}_{r\times n}(\mathbb{F}_{q^m}), \tag{2.5}$$

que suele ser más sencilla de usar.

Comparando las matrices de control (2.5) y (2.3), se deduce que $\Gamma(L,G)$ es un código alternante $\mathcal{A}(\boldsymbol{\alpha},\mathbf{y})$ con $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)$ e $\mathbf{y}=(G(\alpha_1)^{-1},\ldots,G(\alpha_n)^{-1})$. Por lo tanto, $\Gamma(L,G)$ tiene dimensión $k\geq n-mr$ y distancia mínima $d\geq r+1$, donde $r=\deg(G(z))$.

Veamos cómo se expresan los códigos Goppa en función de los códigos Reed-Solomon generalizados.

Teorema 2.3.1. $\Gamma(L,G)$ es la restricción de $GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})$ en $\mathbb{F}_q$, donde $\mathbf{v}=(v_1,\ldots,v_n)$ con $v_i=\frac{G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$, $i=1,\ldots,n$.

Demostración. Sabemos que $\Gamma(L,G)=\mathcal{A}(\boldsymbol{\alpha},\mathbf{y})=GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})^{\perp}\cap\mathbb{F}_q^n$ con $y_i=G(\alpha_i)^{-1}$, $i=1,\ldots,n$.

Vamos a probar que $GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})\subset GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})^{\perp}$ para el caso en el que $y_i=G(\alpha_i)^{-1}$ y $v_i=\frac{G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$, $i=1,\ldots,n$. De aquí, se deduce la igualdad $GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})^{\perp}=GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})$ por tener estos la misma dimensión $k=n-r$. Por lo tanto, intersecando con $\mathbb{F}_q^n$ ambos lados de la igualdad, se obtiene que $\Gamma(L,G)=GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})\cap\mathbb{F}_q^n$, donde $v_i=\frac{G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$, $i=1,\ldots,n$.

Veamos entonces que $GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})\subset GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})^{\perp}$ con $y_i=G(\alpha_i)^{-1}$ y $v_i=\frac{G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$, $i=1,\ldots,n$.

Sea $\mathbf{u}=(u_1,\ldots,u_n)\in GRS_{n-r,n}(\boldsymbol{\alpha},\mathbf{v})$, con $v_i=\frac{G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$, $i=1,\ldots,n$. Entonces, existe un polinomio f de grado menor que $n-r$ tal que

$$u_i=v_if(\alpha_i)=\frac{f(\alpha_i)G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}.$$

Veamos que $\displaystyle\sum_{i=1}^{n}\frac{u_i}{z-\alpha_i}\equiv 0 \mod G(z)$ y entonces se tendrá que $\mathbf{u}\in GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})$, con $y_i=G(\alpha_i)^{-1}$, $i=1,\ldots,n$, puesto que $GRS_{r,n}(\boldsymbol{\alpha},\mathbf{y})$ es también un código Goppa en $\mathbb{F}_{q^m}$, por lo que sus palabras están caracterizadas por la congruencia anterior. Se tiene que

$$\sum_{i=1}^{n}\frac{u_i}{z-\alpha_i}=\sum_{i=1}^{n}\frac{1}{z-\alpha_i}\frac{f(\alpha_i)G(\alpha_i)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}=\frac{1}{\prod_{i=1}^{n}(z-\alpha_i)}\sum_{i=1}^{n}\left(f(\alpha_i)G(\alpha_i)\prod_{j\neq i}\frac{z-\alpha_j}{\alpha_i-\alpha_j}\right).$$

Consideremos $N(z)=\displaystyle\sum_{i=1}^{n}f(\alpha_i)G(\alpha_i)\prod_{j\neq i}\frac{z-\alpha_j}{\alpha_i-\alpha_j}$. Nótese que $\deg(N(z))<n$ y $\deg(f(z)G(z))=\deg(f(z))+\deg(G(z))<(n-r)+r=n$. Además, $N(\alpha_i)=$

$f(\alpha_i)G(\alpha_i)$ para cada $i = 1, \ldots, n$, por lo que el polinomio $N(z) - f(z)G(z)$ tiene n raíces, $\alpha_1, \ldots, \alpha_n$, y es de grado menor que n, luego ha de ser $N(z) = f(z)G(z)$.

Así, se tiene que

$$\sum_{i=1}^{n} \frac{u_i}{z - \alpha_i} = \frac{1}{\prod_{i=1}^{n}(z - \alpha_i)} N(z) = \frac{f(z)G(z)}{\prod_{i=1}^{n}(z - \alpha_i)} \equiv 0 \quad \text{mód } G(z),$$

y se concluye que $GRS_{n-r,n}(\boldsymbol{\alpha}, \mathbf{v}) \subset GRS_{r,n}(\boldsymbol{\alpha}, \mathbf{y})^{\perp}$ como queríamos probar.

□

Teorema 2.3.2. El dual de un código Goppa es $\Gamma(L, G)^{\perp} = T_m(GRS_{r,n}(\boldsymbol{\alpha}, \mathbf{y}))$, donde $y_i = G(\alpha_i)^{-1}$, $i = 1, 2, \ldots, n$.

Demostración. Sabemos que $\Gamma(L, G) = \mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ con $y_i = G(\alpha_i)^{-1}$, $i = 1, 2, \ldots, n$. Además, por el teorema 2.2.1.2, $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})^{\perp} = T_m(GRS_{n-k_0,n}(\boldsymbol{\alpha}, \mathbf{y}))$, de donde se deduce que $\Gamma(L, G)^{\perp} = T_m(GRS_{r,n}(\boldsymbol{\alpha}, \mathbf{y}))$, con $r = n - k_0$ e $y_i = G(\alpha_i)^{-1}$, $i = 1, 2, \ldots, n$. □

2.3.1. Descodificación de un código Goppa

Dado que los códigos Goppa son un caso particular de códigos alternantes, vamos a utilizar el algoritmo 4 AlternantDecoder para construir un algoritmo de descodificación eficiente para los códigos Goppa, el algoritmo 5 GoppaDecoder. Además, del teorema 2.3.1 se sigue que

$$\Gamma(L, G) = \mathcal{A}(\boldsymbol{\alpha}, \mathbf{y}) = GRS_{n-r,n}(\boldsymbol{\alpha}, \mathbf{v}) \cap \mathbb{F}_q^n,$$

donde $\mathbf{v} = (v_1, \ldots, v_n)$ con $v_i = \frac{G(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)}$, $i = 1, ..., n$.

Algorithm 5: GoppaDecoder

Input : $L = \{\alpha_1, \ldots, \alpha_n\}$, el polinomio de Goppa $G(z) \in \mathbb{F}_{q^m}[z]$ y la palabra recibida $\mathbf{r} \in \mathbb{F}_q^n$.

Output: La palabra descodificada $\mathbf{c} \in \Gamma(L, G)$, o **error**.

```
v = (v_1, ..., v_n) con v_i = G(α_i) / ∏_{j≠i}(α_i − α_j), i = 1, ..., n
s = deg(G(z))
if AlternantDecoder(GRS_{s,n}(α, v), s + 1, r)==error then
    return error
else
    return c = AlternantDecoder(GRS_{s,n}(α, v), s + 1, r)
```

El algoritmo 5 GoppaDecoder tiene complejidad cúbica, $O(n^3)$, por ser esta la complejidad computacional del algoritmo 4 AlternantDecoder y por tener el cálculo de $\mathbf{v}$ una complejidad de $O(n^2)$.

Existen otros algoritmos más eficientes para la descodificación de los códigos Goppa, por ejemplo el algoritmo de Patterson que utiliza McEliece en [5] tiene complejidad cuadrática $O(nr)$, pero para nuestros propósitos en criptografía basta con que el algoritmo utilizado tenga complejidad polinómica, por lo que el algoritmo 5 GoppaDecoder es igual de válido.

Capítulo 3

Criptosistema de McEliece

En 1978, R.J. McEliece propuso un criptosistema de clave pública basado en la teoría de códigos algebraicos y construido a partir de un código corrector de errores. Tal criptosistema es el que da nombre a este trabajo: *Criptosistema de McEliece*.

Este criptosistema de clave pública, se apoya en la complejidad de distinguir la estructura de un código aleatorio y en la dificultad de descodificar un código lineal aleatorio, que como veremos en la siguiente sección, se trata de un problema NP-completo, y por lo tanto se tiene que el criptosistema de McEliece pertenece a este grupo de problemas. Después pasaremos a presentar el criptosistema en cuestión y terminaremos el capítulo dando un ejemplo del criptosistema construido a partir de un código Goppa.

3.1. Problemas NP-completos

A continuación vamos a definir lo que es un problema NP-completo y después veremos que descodificar un código sin conocer su estructura es uno de estos problemas. McEliece se basó en esta idea para desarrollar su criptosistema, descrito en la siguiente sección.

Se definen la clase de complejidad P como el conjunto de problemas que se pueden resolver con algoritmos de complejidad polinómica; y la clase de complejidad NP como el conjunto de problemas cuya solución se puede verificar con algoritmos de complejidad polinómica.

Dado que resolver un problema permite verificar una solución dada, se tiene que $P \subset NP$. Se cree que la contención contraria no es cierta, es decir que $P \neq NP$, y este es uno de los problemas del milenio.

Se dice que un problema es NP-completo si pertenece a la clase de complejidad NP y cualquier problema de la clase NP se puede reducir a este con un algoritmo de complejidad polinómica.

Si NP $\neq$ P, entonces ningún problema NP-completo tiene un algoritmo con complejidad polinómica.

Recordemos que si se considera un código lineal sobre $\mathbb{F}_2$ de tipo $[n, k]$, el algoritmo 1 descodificaba por fuerza bruta el código lineal general considerado, y tenía complejidad exponencial, en nuestro caso como estamos considerando un código binario, tendrá complejidad $O(2^k)$.

Problema " Descodificación de un código genérico " :
Dada la palabra recibida $\mathbf{r} \in \mathbb{F}_2^n$ y un código $\mathcal{C} \subset \mathbb{F}_2^n$, encontrar una palabra $\mathbf{c} \in \mathcal{C}$ tal que $d(\mathbf{r}, \mathbf{c}) = \text{mín}\{d(\mathbf{r}, \mathbf{x}) : \mathbf{x} \in \mathcal{C}\}$.

A continuación, vamos a introducir dos problemas que necesitaremos considerar para ver que descodificar un código aleatorio es un problema NP-completo:

Problema " Coset Weights " :
Dado $H \in \mathfrak{M}_{(n-k)\times n}(\mathbb{F}_2)$, $\mathbf{y} \in \mathbb{F}_2^{n-k}$ y w un entero no negativo, decidir si existe un vector $\mathbf{x} \in \mathbb{F}_2^n$ con peso menor o igual que w tal que $\mathbf{y}^t = H\mathbf{x}^t$.

Problema " Three-Dimensional Matching " :
Dado $U \subset T \times T \times T$, donde T es un conjunto finito, decidir si existe un conjunto $W \subset U$ tal que $|W| = |T|$ y tal que ningún par de elementos de W coinciden en ninguna coordenada.

Se sabe que el problema Three-Dimensional Matching es NP-completo, y este se puede reducir mediante un algoritmo de complejidad polinómica al problema Coset Weights, por lo que se deduce que este último también es NP-completo. Por otra parte, el problema de descodificar un código lineal general se puede reducir al problema Coset Weights, también mediante un algoritmo de complejidad polinómica. Por lo tanto, se concluye que el problema de descodificar sin conocer la estructura del código es NP-completo, pero esto no implica que no exista ningún algoritmo con complejidad polinómica, pero lo sugiere fuertemente. Omitimos las demostraciones ya que resultan muy técnicas para este trabajo, para ver la cadena de implicaciones anterior con más profundidad redirigimos al lector a [10].

3.2. Descripción del criptosistema de McEliece

McEliece desarrolló su criptosistema en torno a los códigos Goppa binarios en [5], aunque en este trabajo se expondrá el criptosistema para un código general. La idea es considerar una clase de códigos $\mathcal{C}$ con un algoritmo de descodificación efi-

ciente $D_{\mathcal{C}}$ y camuflar la estructura del código $\mathcal{C}$ de tal forma que parezca un código aleatorio. De esta forma, el criptoanalista se enfrenta a un problema NP-completo, como vimos en la sección anterior.

Sea G una matriz de tamaño $k \times n$ que genera el código $\mathcal{C}$, el cual tiene un algoritmo de descodificación $D_{\mathcal{C}}$ con complejidad polinómica. Codificamos G como $G' = SGP$, donde S es una matriz regular $k \times k$ y P es una matriz de permutación $n \times n$, ambas aleatorias. Obsérvese que como S es invertible, SG es otra matriz generatriz de $\mathcal{C}$, y como P es una matriz monomial por ser de permutación, se tiene que $G' = SGP$ es una matriz generadora de un código equivalente a $\mathcal{C}$.

Los conjuntos de mensajes en texto plano y de mensajes cifrados son $\mathbb{F}_q^k$ y $\mathbb{F}_q^n$ respectivamente, y el conjunto de claves es el conjunto de pares (κ, κ'), con $\kappa \in G_{\mathcal{C}}$ y $\kappa' \in \mathrm{GL}_k(\mathbb{F}_q) \times G_{\mathcal{C}} \times P_n$, donde hemos denotado por $\mathrm{GL}_k(\mathbb{F}_q)$ al conjunto de matrices invertibles $k \times k$ sobre $\mathbb{F}_q$, $G_{\mathcal{C}}$ al conjunto de matrices generatrices de $\mathcal{C}$ y P_n al conjunto de matrices de permutación $n \times n$.

El algoritmo 6 KeysMcEliece genera las claves pública y privada del criptosistema, con complejidad $O(k^2 n)$, ya que este es el coste de calcular SGP, por ser P una matriz de permutación.

Algorithm 6: KeysMcEliece

Input : El código $\mathcal{C}$ sobre $\mathbb{F}_q$ de tipo $[n, k]$.
Output: El par de claves pública y privada (κ, κ') respectivamente.
1 Se elige al azar una matriz generadora de $\mathcal{C}$, G
2 Se eligen al azar una matriz $k \times k$ regular, S, y una matriz de permutación $n \times n$, P
3 Se calcula $G' = SGP$
4 $\kappa = G'$
5 $\kappa' = (S, G, P)$
6 **return** (κ, κ')

Una vez generadas las claves pública $\kappa = G'$ y privada $\kappa' = (S, G, P)$, vamos a describir los procesos de cifrado y descifrado del criptosistema de McEliece.

La aplicación de cifrado $c : \mathbb{F}_q^k \times G_{\mathcal{C}} \longrightarrow \mathbb{F}_q^n$ viene dada por

$$c(\mathbf{m}, G') = \mathbf{m}G' + \mathbf{e},$$

donde $\mathbf{e} \in \mathbb{F}_q^n$ se elige al azar y tiene peso a lo sumo t, donde t es la capacidad correctora de $\mathcal{C}$. El algoritmo 7 CifMcEliece muestra el proceso de cifrado en forma algorítmica y tiene complejidad computacional $O(kn)$.

Algorithm 7: CifMcEliece

Input : El mensaje en texto plano $\mathbf{m} \in \mathbb{F}_q^k$ y la clave pública $\kappa = G'$.
Output: El mensaje cifrado $\mathbf{c} \in \mathbb{F}_q^n$.

1 Se elige al azar $\mathbf{e} \in \mathbb{F}_q^n$ con peso menor o igual que t
2 Se calcula $\mathbf{c} = \mathbf{m}G' + \mathbf{e}$
3 **return c**

La aplicación de descifrado $d : \mathbb{F}_q^n \times (\mathrm{GL}_k(\mathbb{F}_q) \times G_{\mathcal{C}} \times P_n) \longrightarrow \mathbb{F}_q^k$ viene dada por

$$d(\mathbf{c}, (S, G, P)) = \mathrm{Gauss}_G \left(D_{\mathcal{C}}(\mathbf{c}P^{-1}) \right) S^{-1},$$

donde Gauss_G indica que se resuelve un sistema de ecuaciones con matriz G y $D_{\mathcal{C}} : \mathbb{F}_q^n \longrightarrow \mathcal{C}$ es un algoritmo de descodificación eficiente para $\mathcal{C}$. El algoritmo 8 DescifMcEliece muestra el proceso de descifrado en forma algorítmica y su complejidad será en general la misma que la de $D_{\mathcal{C}}$, ya que $D_{\mathcal{C}}$ tendrá complejidad al menos $O(n^3)$ y el resto de operaciones del algoritmo tinen complejidad $O(n^3)$. En resumen, el algoritmo 8 DescifMcEliece tiene complejidad polinómica.

Algorithm 8: DescifMcEliece

Input : El mensaje cifrado $\mathbf{c} \in \mathbb{F}_q^n$ y la clave privada $\kappa' = (S, G, P)$.
Output: El mensaje original $\mathbf{m} \in \mathbb{F}_q^k$.

1 Se calcula $\mathbf{c}_1 = \mathbf{c}P^{-1}$
2 Se calcula $\mathbf{c}_2 = D_{\mathcal{C}}(\mathbf{c}_1) = \mathbf{m}SG$
3 Se calcula $\mathbf{c}_3 = \mathbf{m}S$, resolviendo el sistema de ecuaciones $\mathbf{c}_2 = \mathbf{c}_3 G$
4 Se calcula $\mathbf{c}_4 = \mathbf{c}_3 S^{-1}$
5 **return** $\mathbf{c}_4$

Veamos que efectivamente esto define un criptosistema, viendo que se cumple que $d(c(\mathbf{m}, G'), (S, G, P)) = \mathbf{m}$, para cada $\mathbf{m} \in \mathbb{F}_q^k$, y para cada par de claves $(G', (S, G, P)) \in G_{\mathcal{C}} \times (\mathrm{GL}_k(\mathbb{F}_q) \times G_{\mathcal{C}} \times P_n)$, donde $G' = SGP$:

$$\begin{aligned} d(c(\mathbf{m}, G'), (S, G, P)) &= d(\mathbf{m}G' + \mathbf{e}, (S, G, P)) \\ &= \mathrm{Gauss}_G \left(D_{\mathcal{C}}(\mathbf{m}SG + \mathbf{e}P^{-1}) \right) S^{-1} \\ &= \mathrm{Gauss}_G \left(\mathbf{m}SG \right) S^{-1} \\ &= (\mathbf{m}S)S^{-1} \\ &= \mathbf{m}. \end{aligned}$$

Nótese que el algoritmo de descodificación $D_{\mathcal{C}}$ descodifica el mensaje $\mathbf{m}SG + \mathbf{e}P^{-1}$ correctamente, puesto que $\mathbf{e}P^{-1}$ tiene el mismo peso que $\mathbf{e}$, es decir, menor o igual que t, la capacidad correctora de $\mathcal{C}$.

Por último, vamos a comprobar que se satisfacen las condiciones de Diffie-Hellman, enumeradas en la sección 1.1. Códigos Criptográficos, ya que estas son propiedades que todo criptosistema de clave pública debe cumplir, y por tanto también el criptosistema de McEliece debe verificarlas:

1. La generación del par de claves pública y privada $(G', (S, G, P))$ es computacionalmente sencilla debido a que el algoritmo 6 KeysMcEliece tiene complejidad polinómica.

2. Conocida la clave pública $G' = SGP$, el proceso de cifrado es computacionalmente sencillo, ya que la aplicación de cifrado se puede implementar con el algoritmo 7 CifMcEliece que tiene complejidad polinómica.

3. Como la aplicación de descifrado se puede implementar con el algoritmo 8 DescifMcEliece que tiene complejidad polinómica, se tiene que conocida la clave privada (S, G, P), el proceso de descifrado es computacionalmente sencillo.

4. Dado que el problema de descodificar un código lineal aleatorio es NP-completo, se sigue que es computacionalmente difícil obtener el mensaje original a partir del mensaje cifrado sin conocer la clave privada.

5. Obtener la clave privada (S, G, P) conocida la clave pública $G' = SGP$, es computacionalmente difícil, debido a que la factorización de un producto de matrices cualquiera tiene complejidad exponencial.

Podemos concluir que desde el marco teórico, el criptosistema de McEliece es un gran candidato a criptosistema de clave pública. Sin embargo, es difícil de aplicar en la práctica debido al gran tamaño de las claves. Nótese que si G es una matriz generadora de $\mathcal{C}$ y la multiplicamos por una matriz invertible S elegida uniformemente al azar, entonces SG es una matriz generadora de $\mathcal{C}$ elegida uniformemente al azar. Por ello,

$$|G_{\mathcal{C}}| = |GL_k(\mathbb{F}_q)| = (q^k - 1)(q^k - q)(q^k - q^2)\dots(q^k - q^{k-1}) = O(q^{k^2}).$$

Además, $|P_n| = n!$, por ser los elementos de P_n permutaciones de columnas.

Por lo tanto,

$$|\mathcal{K}| = |G_{\mathcal{C}} \times (GL_k(\mathbb{F}_q) \times G_{\mathcal{C}} \times P_n)| = |GL_k(\mathbb{F}_q)|^3 |P_n| = O(q^{3k^2})n!,$$

es decir, el orden del tamaño del conjunto de claves es polinómico en $q^k = |\mathbb{F}_q^k| = |\mathcal{M}|$, el tamaño del mensaje, y por consiguiente, mayor que el tamaño del conjunto de claves de los criptosistemas de clave pública que se utilizan hoy en día como veremos a continuación. Este inconveniente se intentó reducir introduciendo una estructura adicional a los códigos utilizados, pero esto les hizo a su vez más vulnerables, pues les dotaba de debilidades esenciales.

Veamos cuál es el tamaño del conjunto de claves de los criptosistemas de clave pública más utilizados actualmente, el RSA y los basados en el problema del logaritmo discreto como ElGamal, para compararlos con el tamaño del conjunto de claves del criptosistema de McEliece.

Las claves del criptosistema RSA se generan como se muestra en el algoritmo 9 KeysRSA.

Algorithm 9: KeysRSA

Output: El par de claves pública y privada (κ, κ') respectivamente.

1 Se eligen al azar dos números primos p y q distintos
2 Se calculan $n = pq$ y $\varphi(n) = (p-1)(q-1)$, donde φ es la función indicatriz de Euler
3 Se escoge al azar un entero e tal que $0 < e < \varphi(n)$ y $\gcd(e, \varphi(n)) = 1$
4 Se calcula d tal que $d \equiv e^{-1} \mod \varphi(n)$
5 $\kappa = (n, e)$
6 $\kappa' = d$
7 **return** (κ, κ')

Como $e, d < n$, el tamaño del conjunto de claves es menor que $3n$, con lo que $|\mathcal{K}| = O(n)$, es decir, el orden del tamaño del conjunto de claves es lineal en $n = |\mathbb{Z}_n| = |\mathcal{M}|$, el tamaño del mensaje.

Por otro lado, las claves del criptosistema de ElGamal se generan como se muestra en el algoritmo 10 KeysElGamal.

Algorithm 10: KeysElGamal

Input : Un grupo cíclico finito $\mathcal{G}$ generado por g
Output: El par de claves pública y privada (κ, κ') respectivamente.

1 Se elige al azar $n \in \mathbb{Z}_N \setminus \{0, 1\}$, donde $N = |G|$
2 $\kappa = (\mathcal{G}, g, g^n)$
3 $\kappa' = n$
4 **return** (κ, κ')

Como el conjunto de generadores del grupo cíclico y $n \in \mathbb{Z}_N \setminus \{0, 1\}$ son menores que $N = |\mathcal{G}|$, se tiene que el tamaño del conjunto de claves es menor que $3N$, luego $|\mathcal{K}| = O(N)$, es decir, el orden del tamaño del conjunto de claves es lineal en $N = |\mathcal{G}| = |\mathcal{M}|$, el tamaño del mensaje.

En consecuencia, como el tamaño del conjunto de claves del criptosistema RSA y ElGamal es lineal en el tamaño del mensaje, y por lo tanto mucho menor que el tamaño del conjunto de claves del criptosistema de McEliece, que es polinómico en el tamaño del mensaje, se seguirán utilizando estos criptosistemas de clave pública frente al de McEliece.

3.3. Ejemplo

En esta sección vamos a presentar un ejemplo académico de un código Goppa en el que se calcularán sus parámetros fundamentales y unas matrices generadora y de control. Después, se describirá el criptosistema de McEliece construido a partir de ese código Goppa. Además, para este criptosistema, se codificará un mensaje al que después añadiremos un error y por último veremos que al descodificar el mensaje encriptado obtendremos el mensaje original.

Consideramos el código Goppa $\Gamma(L, G)$ sobre $\mathbb{F}_{2^3}$, con polinomio de Goppa $G(z) = z^2 + z + 1$ y $L = \mathbb{F}_{2^3} = \{0, 1, \alpha, \ldots, \alpha^6\}$, donde α es un elemento primitivo de $\mathbb{F}_{2^3} \setminus \{0\}$. Con estos datos ya podemos decir que $\Gamma(L, G)$ tiene longitud $n = |L| = |\mathbb{F}_{2^3}| = 8$, dimensión $k \geq 8 - 3 \cdot 2 = 2$ y distancia mínima $d \geq 2 + 1 = 3$.

Primero, vamos a reescribir el conjunto $\mathbb{F}_{2^3}$ para que sea más fácil comprobar que efectivamente ningún elemento de L es raíz del polinomio de Goppa. Podemos escribir $\mathbb{F}_{2^3} = \mathbb{F}_2[z]/(f)$, donde $f(z) = z^3 + z + 1$ es un polinomio irreducible de $\mathbb{F}_2[z]$ y $\deg(f) = 3$.

Podemos escoger $\alpha \in \mathbb{F}_{2^3}$ como una raíz de $f(z)$ (nótese que $\mathbb{F}_{2^3}$ es una extensión de $\mathbb{F}_2$), entonces se cumple que $\alpha^3 = \alpha + 1$. Se tiene que tal α es un elemento primitivo de $\mathbb{F}_{2^3}$ ya que

$$\begin{aligned}
\alpha^4 &= \alpha^3\alpha = (\alpha + 1)\alpha = \alpha^2 + \alpha, \\
\alpha^5 &= \alpha^4\alpha = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1, \\
\alpha^6 &= \alpha^5\alpha = \alpha^3 + \alpha^2 + \alpha = \alpha + 1 + \alpha^2 + \alpha = \alpha^2 + 1, \\
\alpha^7 &= \alpha^6\alpha = \alpha^3 + \alpha = \alpha + 1 + \alpha = 1.
\end{aligned}$$

Entonces podemos expresar también $\mathbb{F}_{2^3}$ como

$$\mathbb{F}_{2^3} = \left\{0, 1, \alpha, \alpha^2, \alpha + 1, \alpha^2 + \alpha, \alpha^2 + \alpha + 1, \alpha^2 + 1\right\}.$$

Comprobemos que efectivamente los elementos de L no son raíces del polinomio de Goppa $G(z) = z^2 + z + 1$, y entonces tendrá sentido calcular $G(\alpha_i)^{-1}$ para cada $\alpha_i \in L$:

$$
\begin{aligned}
G(0) &= 1 \implies G(0)^{-1} = 1,\\
G(1) &= 1 \implies G(1)^{-1} = 1,\\
G(\alpha) &= \alpha^2+\alpha+1 = \alpha^5 \implies G(\alpha)^{-1} = \alpha^2,\\
G(\alpha^2) &= \alpha^4+\alpha^2+1 = \alpha+1 = \alpha^3 \implies G(\alpha^2)^{-1} = \alpha^4,\\
G(\alpha^3) &= \alpha^6+\alpha^3+1 = \alpha^2+\alpha+1 = \alpha^5 \implies G(\alpha^3)^{-1} = \alpha^2,\\
G(\alpha^4) &= \alpha^8+\alpha^4+1 = \alpha+\alpha^4+1 = \alpha^2+1 = \alpha^6 \implies G(\alpha^4)^{-1} = \alpha,\\
G(\alpha^5) &= \alpha^{10}+\alpha^5+1 = \alpha^3+\alpha^5+1 = \alpha^2+1 = \alpha^6 \implies G(\alpha^5)^{-1} = \alpha,\\
G(\alpha^6) &= \alpha^{12}+\alpha^6+1 = \alpha^5+\alpha^6+1 = \alpha+1 = \alpha^3 \implies G(\alpha^6)^{-1} = \alpha^4.
\end{aligned}
$$

Sabemos que una matriz de control de $\Gamma(L,G)$ viene dada por la expresión (2.5), y como $\deg(G(z)) = 2$ se tiene que una matriz de control del código es:

$$
H = \begin{pmatrix} 1 & 1 & \alpha^2 & \alpha^4 & \alpha^2 & \alpha & \alpha & \alpha^4 \\ 0 & 1 & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^3 \end{pmatrix} \in \mathfrak{M}_{2\times 8}(\mathbb{F}_{2^3}).
$$

Si consideramos la base $\{1,\alpha,\alpha^2\}$ de $\mathbb{F}_{2^3}$ sobre $\mathbb{F}_2$, se tiene que sobre esta base,

$$
\mathbb{F}_{2^3} = \{(0,0,0)^t,(1,0,0)^t,(0,1,0)^t,(0,0,1)^t,(1,1,0)^t,(0,1,1)^t,(1,1,1)^t,(1,0,1)^t\},
$$

por lo que otra matriz de control del código es

$$
\widetilde{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \in \mathfrak{M}_{6\times 8}(\mathbb{F}_2).
$$

Sabemos que si $\widetilde{H}$ es una matriz de control del código Goppa, entonces G será una matriz generadora del código si cumple que $\widetilde{H}G^t = 0$. Por lo tanto, para obtener G vamos a calcular el núcleo de $\widetilde{H}$ módulo 2:

$$
\begin{aligned}
\ker \widetilde{H} &= \ker \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}\\
&= \langle\{g_1,\ldots,g_8 \in \mathbb{F}_2 : g_1 = g_2, g_4 = g_3, g_5 = g_2+g_3, g_6 = g_3, g_7 = g_2+g_3, g_8 = g_2+g_3\}\rangle\\
&= \langle\{(g_2,g_2,g_3,g_3,g_2+g_3,g_3,g_2+g_3,g_2+g_3) : g_2,g_3 \in \mathbb{F}_2\}\rangle\\
&= \langle\{(1,1,0,0,1,0,1,1),(0,0,1,1,1,1,1,1)\}\rangle,
\end{aligned}
$$

donde la primera igualdad se ha obtenido de hacer operaciones elementales en $\mathbb{F}_2$ sobre las filas de $\widetilde{H}$, obteniendo así una matriz equivalente, por lo que ambas matrices tienen el mismo núcleo.

Por lo tanto, la matriz

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \in \mathfrak{M}_{2\times 8}(\mathbb{F}_2),$$

es una matriz generadora del código Goppa, y además, como esta ha de tener k filas y n columnas, siendo n y k la longitud y dimensión del código respectivamente, se deduce que $k = 2$.

En este ejemplo es fácil calcular todas las palabras de $\Gamma(L, G)$ a partir de la matriz generadora, porque al ser las operaciones sobre $\mathbb{F}_2$, la única palabra no nula del código que nos queda es la que se obtiene al sumar las dos filas de G, luego

$$\Gamma(L, G) = \{(0,0,0,0,0,0,0,0), (1,1,0,0,1,0,1,1), (0,0,1,1,1,1,1,1), (1,1,1,1,0,1,0,0)\},$$

de donde se deduce que la distancia mínima es $d = 5$ por ser este el menor peso de Hamming de las palabras no nulas del código, y por lo tanto, la distancia mínima supera la cota general, que habíamos visto que era $d \geq 3$.

Se tiene así que $\Gamma(L, G)$ es de tipo $[8, 2, 5]$ y por tanto corrige hasta $t = \lfloor \frac{5-1}{2} \rfloor = 2$ errores.

A continuación pasamos a describir el criptosistema de McEliece definido a partir del código Goppa anterior.

Para elegir aleatoriamente las matrices $S \in GL_2(\mathbb{F}_2)$ y $P \in P_8$ se han implementado las líneas de código que aparecen en la figura 3.1 en SageMath, obteniendo así

$$S = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

La clave privada del criptosistema será la terna $(S, G, P) \in GL_2(\mathbb{F}_2) \times G_{\Gamma(L,G)} \times P_8$ y la clave pública vendrá dada por la matriz

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

```
#se elige al azar una matriz invertible S 2x2
S=random_matrix(GF(2),2,2);
while det(S)==0 :
    S=random_matrix(GF(2),2,2);
S
```

```
[1 1]
[0 1]
```

```
#se elige al azar una matriz de permutación P 8x8
P=zero_matrix(8,8);
position=Permutations(8).random_element();
for i in srange(0,8) :
    P[i,position[i]-1]=1;
P=matrix(GF(2),P);P
```

```
[0 0 0 0 1 0 0 0]
[0 0 1 0 0 0 0 0]
[0 0 0 1 0 0 0 0]
[0 0 0 0 0 0 0 1]
[0 1 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 1 0]
```

Figura 3.1: Generación aleatoria de las matrices S y P en SageMath.

Supongamos ahora que queremos enviar el mensaje $\mathbf{m} = (1,1) \in \mathbb{F}_2^2$, para ello, utilizamos la aplicación de cifrado obteniendo

$$c(\mathbf{m}, G') = \mathbf{m}G' + \mathbf{e} = (0,1,1,0,1,1,1,0) + (0,0,1,0,0,0,0,1) = (0,1,0,0,1,1,1,1),$$

donde $\mathbf{e} = (0,0,1,0,0,0,0,1) \in \mathbb{F}_2^8$ se ha generado aleatoriamente (teniendo en cuenta que $w(\mathbf{e}) \leq 2$) con SageMath como se muestra en la figura 3.2.

Por último, vamos a descodificar el mensaje cifrado $\mathbf{c} = (0,1,0,0,1,1,1,1) \in \mathbb{F}_2^8$ utilizando la clave privada $(S, G, P) \in GL_2(\mathbb{F}_2) \times G_{\Gamma(L,G)} \times P_8$, para obtener el mensaje original $\mathbf{m} = (1,1)$. Procederemos como en el algoritmo 8 DescifMcEliece.

Primero calculamos

$$\mathbf{c}_1 = \mathbf{c}P^{-1} = (0,1,0,0,1,1,1,1) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} = (1,0,0,1,1,0,1,1).$$

```
#Generamos el vector de error e con peso menor o igual que 2 aleatoriamente
e=[0,0,0,0,0,0,0,0];
cont=0;
for i in srange(0,8):
    if cont<2 :
        e[i]=randint(0,1);
        if e[i]==1:
            cont=cont+1;
e=matrix(GF(2),e);e
```

```
[0 0 1 0 0 0 0 1]
```

Figura 3.2: Generación aleatoria del error **e** en SageMath.

Utilizando el algoritmo 5 GoppaDecoder como nuestro algoritmo de descodificación eficiente $D_{\Gamma(L,G)}$, corregimos los errores que se han producido, obteniendo

$$\mathbf{c}_2 = D_{\Gamma(L,G)}(\mathbf{c}_1) = D_{\Gamma(L,G)}((1,0,0,1,1,0,1,1)) = (1,1,0,0,1,0,1,1).$$

A continuación, para calcular $\mathbf{c}_3$ resolvemos el sistema de ecuaciones lineales $\mathbf{c}_2 = \mathbf{c}_3 G$. Nótese que $\mathbf{c}_2$ es la primera fila de G, luego

$$\mathbf{c}_3 = (1,0).$$

Por último, obtenemos que el mensaje original es

$$\mathbf{m} = \mathbf{c}_3 S^{-1} = (1,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (1,1).$$

Capítulo 4

Ataques al criptosistema de McEliece

Pasamos ahora a describir los dos tipos de ataques que hay para intentar romper los criptosistemas construidos a partir de códigos, particularizándolo al criptosistema de McEliece.

Supongamos que un criptoanalista quiere determinar el mensaje en texto plano $\mathbf{m}$, conociendo la clave pública G' e interceptando el mensaje codificado $\mathbf{c}$. Para ello, tiene dos ataques básicos posibles:

1. Ataques de descodificación genéricos: intentan recuperar $\mathbf{m}$ a partir de $\mathbf{c}$ utilizando G', sin conocer S, G y P. Por lo tanto, este ataque trata de descodificar un código lineal que parece aleatorio, que hemos visto que es un problema NP-completo, por lo que si los parámetros del código son suficientemente grandes, se tiene complejidad exponencial.

2. Ataques estructurales: buscan recuperar S, G y P a partir de G' para poder descodificar con $D_{\mathcal{C}}$. Aunque los parámetros n y t sean muy grandes, y por lo tanto se tengan muchas posibilidades para G, sin contar con las posibilidades que hay para S y P, si el código utilizado para describir el criptosistema tiene alguna propiedad que se pueda observar en la mayoría de las matrices generatrices, como por ejemplo la ciclicidad en los códigos cíclicos, entonces estos ataques son efectivos.

En [5], se estudian los dos tipos de ataques mencionados para los códigos Goppa binarios de tipo $[1024, 524]$, que son capaces de corregir 50 errores. Se esboza que ambos ataques tienen una complejidad exponencial, y por lo tanto no son útiles en la práctica. Sin embargo, en las siguientes secciones de este capítulo, vamos a profundizar un poco más en estos dos tipos de ataques, dando un ejemplo de algoritmo que hace que el ataque sea efectivo dependiendo de los parámetros o la estructura del código utilizado en el criptosistema de McEliece.

4.1. Ataques de descodificación genéricos

Vamos a obtener estimaciones para la complejidad de este tipo de ataques. Los más eficientes se obtuvieron implementando mejoras del algoritmo ISD (Information Set Decoding).

Primero describiremos cómo el algoritmo 11 ISD descodifica la palabra recibida, encontrando las posiciones en las que no se ha producido un error, localizando así los errores. Después recrearemos el algoritmo mencionado, y a continuación, se presentará la primera mejora del mismo, propuesta en [3] por Lee y Brickell en 1988.

Definición 4.1.1. Se define un conjunto de información de un código $\mathcal{C}$ de tipo $[n, k]$ como un subconjunto de índices de $\{1, 2, \ldots, n\}$ de tamaño k tal que para todo $\mathbf{m} \in \mathbb{F}_q^k$, existe un único $\mathbf{c} \in \mathcal{C}$ cumpliendo que $\mathbf{c}_I = \mathbf{m}_I$, donde $\mathbf{c}_I$ y $\mathbf{m}_I$ son las proyecciones de $\mathbf{c}$ y $\mathbf{m}$ sobre las coordenadas en I, es decir,

$$I = \left\{ i_1, \ldots, i_k : \forall \mathbf{m} \in \mathbb{F}_q^k, \exists! \mathbf{c} \in \mathcal{C} \text{ con } c_{i_j} = m_{i_j}, \forall j = 1, \ldots, k \right\} \subset \{1, 2, \ldots, n\}.$$

Como otras veces, consideramos $\mathcal{C}$ un código lineal de tipo $[n, k]$ generado por una matriz G, $\mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^k$ la palabra recibida, donde $\mathbf{c} \in \mathcal{C}$ y $\mathbf{e} \in \mathbb{F}_q^k$ es el error producido.

Sea I un conjunto de información de $\mathcal{C}$, entonces existe un único $\mathbf{c} \in \mathcal{C}$ tal que $\mathbf{r}_I = \mathbf{c}_I$. Consideremos la submatriz G_I de G, formada por las columnas i_j-ésimas de G para todo $i_j \in I$.

Veamos que por ser G_I regular, por la construcción de I, se tiene que la única solución de $\mathbf{m}G = \mathbf{c}$ es $\mathbf{m} = \mathbf{r}_I G_I^{-1} \in \mathbb{F}_q^k$, si $\mathbf{e}_I = \mathbf{0}$:

$$\mathbf{m}G = \mathbf{c} \Rightarrow \mathbf{m}G_I = \mathbf{c}_I \Rightarrow \mathbf{m} = \mathbf{c}_I G_I^{-1} = \mathbf{r}_I G_I^{-1} \Rightarrow \mathbf{m} = \mathbf{r}_I G_I^{-1},$$

donde en la primera implicación hemos restringido la igualdad a I. Además, la solución encontrada es única por ser G_I invertible y ser $\mathbf{c}$ la única palabra de $\mathcal{C}$ tal que $\mathbf{r}_I = \mathbf{c}_I$.

Consideremos $\widetilde{G} = G_I^{-1} G$. Como $\mathbf{m} = \mathbf{r}_I G_I^{-1} \in \mathbb{F}_q^k$ es la única solución de $\mathbf{m}G = \mathbf{c}$ si $\mathbf{e}_I = \mathbf{0}$, se descodifica $\mathbf{r}$ como $\mathbf{c} = \mathbf{r}_I G_I^{-1} G = \mathbf{r}_I \widetilde{G}$.

La probabilidad de que la descodificación dada por el algoritmo 11 ISD sea correcta, es decir, la probabilidad de que no haya ningún error en los k subíndices elegidos al azar, entre las n coordenadas con t errores es $p_{ISD} = \frac{\binom{n-t}{k}}{\binom{n}{k}}$, donde t es el número de errores que se pueden corregir.

Algorithm 11: Information Set Decoding (ISD)

Input : Una matriz generadora G del código $\mathcal{C}$ de tipo $[n, k]$, la palabra recibida $\mathbf{r}$ y una colección $\mathcal{I}(\mathcal{C})$ de todos los conjuntos de información del código $\mathcal{C}$.

Output: La palabra del código $\mathbf{c} \in \mathcal{C}$ tal que $d(\mathbf{r}, \mathbf{c}) = d(\mathbf{r}, \mathcal{C})$.

```
Sea c = 0
for I ∈ I(C) do
    G' = G_I^{-1} G
    c' = r_I G'
    if d(c', r) < d(c, r) then
        c = c'
return c
```

Lee y Brickell propusieron la primera mejora del algoritmo ISD, permitiendo que en el conjunto I se produzcan un número pequeño de errores p. Así, se admite que $\mathbf{r}_I = \mathbf{c}_I + \mathbf{e}_I$ con $w(\mathbf{e}_I) \leq p$. En el algoritmo 12 Lee-Brickell ISD se implementa dicha mejora.

Algorithm 12: Lee-Brickell ISD

Input : Una matriz generadora G del código t-corrector $\mathcal{C}$ de tipo $[n, k]$, la palabra recibida $\mathbf{r}$, el número de intentos N_{trials} y un parámetro p.

Output: La palabra del código $\mathbf{c} \in \mathcal{C}$ tal que $d(\mathbf{r}, \mathbf{c}) \leq t$ ó **error**.

```
Sean c = 0 y N_tr = 0
while N_tr < N_trials do
    N_tr = N_tr + 1
    Se elige al azar un subconjunto I de {1, ..., n} de tamaño k
    if det(G_I) ≠ 0 then
        G̃ = G_I^{-1} G
        for e_I of w(e_I) ≤ p do
            c̃ = (r_I + e_I) G̃
            if d(c̃, r) ≤ t then
                return c̃
return error
```

La probabilidad de que la descodificación dada por el algoritmo 12 sea correcta con un número de intentos $N_{trial} = 1$ es $p_{LB} = \frac{\binom{t}{p}\binom{n-t}{k-p}}{\binom{n}{k}}$, que es mayor que la pro-

babilidad p_{ISB} obtenida con el algoritmo ISD. Nótese que el bucle for del algoritmo 12 se realizará 2^p veces, por lo que si queremos que el algoritmo sea eficiente, p ha de ser pequeño.

El algoritmo debido a Lee y Brickell es una de las mejoras que ha experimentado el ISD y que resulta ser eficiente como ataque para romper la propuesta original de McEliece utilizando códigos Goppa binarios de tipo $[1024, 524]$. Dado que se trata de un ataque de descodificación genérico y no utiliza la estructura particular de los códigos Goppa, es posible garantizar la seguridad del criptosistema, aumentando los parámetros del código Goppa.

4.2. Ataques estructurales

Como comentamos anteriormente, este tipo de ataque intenta deducir la estructura del código utilizado para construir el criptosistema a partir del código generado por la clave pública G'. Actualmente, los códigos Goppa binarios son uno de los pocos códigos que han resistido a este tipo de ataque. Aunque se han propuesto muchos otros códigos que tienen un algoritmo de descodificación eficiente, la mayoría de ellos son vulnerables a los ataques estructurales, por ejemplo, los códigos BCH, los códigos Reed-Muller y los códigos Reed-Solomon generalizados entre otros. A continuación, veremos un ataque que consigue recuperar la estructura de los códigos Reed-Solomon generalizados con complejidad polinómica, por lo tanto, queda probado que el criptosistema de McEliece no es seguro si se construye a partir de códigos Reed-Solomon generalizados.

Por otro lado, se puede concluir que el criptosistema de McEliece es seguro si se define a partir de códigos Goppa, ya que es difícil distinguir su estructura de la de un código aleatorio. Aunque en la literatura no haya ninguna referencia en la que se demuestre esto formalmente, podemos decir que el criptosistema construido a partir de códigos Goppa será seguro hasta que se diseñe un ataque estructural que consiga obtener la estructura de este tipo de códigos.

4.2.1. Ataques con códigos Reed-Solomon generalizados

Ya hemos visto que uno de los pilares en los que se apoya el criptosistema de McEliece es que la estructura del código utilizado ha de parecer aleatoria. Sin embargo, el producto componente a componente utilizado en la descripción de los cógidos Reed-Solomon generalizados, hace que estos se puedan identificar, lo que implica que su utilización en el criptosistema de McEliece lo vuelva vulnerable.

Vamos a describir el ataque que propusieron Sidelnikov y Shestakov en [8] en el año 1992, que rompe el criptosistema de McEliece en tiempo polinómico, si se

utilizan códigos Reed-Solomon generalizados para describir el criptosistema. Tal algoritmo calcula la matriz regular S y los parámetros $\mathbf{x}$ y $\mathbf{z}$, que describen el código $GRS_{n-k,n}(\mathbf{x},\mathbf{z})$ con matriz de control GP, a partir de la clave pública $G' = SGP \in \mathfrak{M}_{k\times n}(\mathbb{F}_q)$.

Nota. Si G es una matriz generadora del código $GRS_{k,n}(\boldsymbol{\alpha},\mathbf{v})$ y P es una matriz de permutación $n \times n$, entonces GP es una matriz generatriz de $GRS_{k,n}(\mathbf{x},\mathbf{y})$, que es un código equivalente al generado por G. Por lo tanto, también GP es una matriz de control de $GRS_{k,n}(\mathbf{x},\mathbf{y})^{\perp} = GRS_{n-k,n}(\mathbf{x},\mathbf{z})$, donde $\mathbf{x},\mathbf{z} \in \mathbb{F}_q^n$.

Una vez obtenidos la matriz regular S y los parámetros $\mathbf{x}$, $\mathbf{z}$ (con los que podemos construir una matriz generadora GP de $GRS_{n-k,n}(\mathbf{x},\mathbf{z})^{\perp}$, por tener estos códigos unas matrices generadora y de control de una forma específica) es fácil descodificar el mensaje interceptado $\mathbf{c}$ haciendo: $Gauss\left(D_{\mathcal{C}}(\mathbf{c})\right)S^{-1}$, ya que

$$\begin{aligned} Gauss\left(D_{\mathcal{C}}(\mathbf{c})\right)S^{-1} &= Gauss\left(D_{\mathcal{C}}(\mathbf{m}SGP+\mathbf{e})\right)S^{-1} = Gauss\left(\mathbf{m}SGP\right)S^{-1} \\ &= (\mathbf{m}S)S^{-1} = \mathbf{m}. \end{aligned}$$

Supongamos conocida la clave pública $G' = SGP \in \mathfrak{M}_{k\times n}(\mathbb{F}_q)$ del criptosistema de McEliece.

Vamos a denotar por $A = A(x_1,\ldots,x_n;z_1,\ldots,z_n) = GP$ a la matriz de control

$$A = \begin{pmatrix} z_1 & z_2 & \ldots & z_n \\ z_1x_1 & z_2x_2 & \ldots & z_nx_n \\ \vdots & \vdots & \ddots & \vdots \\ z_1x_1^{r-1} & z_2x_2^{r-1} & \ldots & z_nx_n^{r-1} \end{pmatrix},$$

donde $r = n-k$, $x_i \in \mathbb{F}_q \cup \{\infty\}$, $1 \le i \le n$, $x_i \neq x_j$ para todo $i \neq j$, $z_i \in \mathbb{F}_q \setminus \{0\}$ y si $x_j = \infty$ entonces la correspondiente columna se define como el vector $(0,\ldots,0,z_j)^t$.

Podemos suponer que la matriz G es de la forma anterior, y entonces, por ser P una matriz de permutación, se tiene que $A = GP$ también es de esa forma. Si la estructura de G no fuera así, como la matriz S es escoge uniformemente al azar, podemos encontrar una matriz $\widehat{S}$ invertible tal que $\widehat{G} = \widehat{S}^{-1}SG$ es de la forma deseada.

Además, $G' = SGP = SA$ es de la forma

$$G' = \begin{pmatrix} z_1f_1(x_1) & z_2f_1(x_2) & \ldots & z_nf_1(x_n) \\ z_1f_2(x_1) & z_2f_2(x_2) & \ldots & z_nf_2(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ z_1f_r(x_1) & z_2f_r(x_2) & \ldots & z_nf_r(x_n) \end{pmatrix},$$

donde $f_j(x)$, $j = 1,\ldots,r$ son polinomios linealmente independientes sobre $\mathbb{F}_q$ de grado menor que r, que vienen determinados por la matriz S, donde $f_j(\infty)$ es el

coeficiente de x^{r-1}.

Suponiendo conocida G', vamos a calcular $x_1, \ldots, x_n \in \mathbb{F}_q \cup \{\infty\}$, $z_1, \ldots, z_n \in \mathbb{F}_q \setminus \{0\}$ y una matriz regular S tal que

$$G' = SA(x_1, \ldots, x_n; z_1, \ldots, z_n). \tag{4.1}$$

Procederemos a resolver el problema en dos etapas. En la primera de ellas calcularemos $x_1, \ldots, x_n \in \mathbb{F}_q \cup \{\infty\}$ y en la segunda etapa encontraremos $z_1, \ldots, z_n \in \mathbb{F}_q \setminus \{0\}$, junto con la matriz S.

Antes de empezar con la primera etapa vamos hacer algunas anotaciones, que nos servirán para justificar las operaciones realizadas en el algoritmo 13.

Sean $(S; x_1, \ldots, x_n; z_1, \ldots, z_n)$ una solución de la ecuación (4.1), $a, b \in \mathbb{F}_q$, con $a \neq 0$, y sean $s_{ij} \in \mathbb{F}_q$, $0 \leq i, j \leq r-1$ una solución no nula del sistema lineal

$$(ax+b)^i = \sum_{j=0}^{r-1} s_{ij} x^j, \quad 0 \leq i \leq r-1. \tag{4.2}$$

Consideramos

$$S_1 = (s_{ij})_{0 \leq i,j \leq r-1}, \quad \text{y} \quad d_j = \begin{cases} 1 & \text{si} \quad x_j \neq \infty \\ a^{-(r-1)} & \text{si} \quad x_j = \infty \end{cases}, \; 1 \leq j \leq n.$$

Nótese que por definición S_1 es una matriz triangular inferior sin ceros en la diagonal, y además $s_{r-1,r-1} = a^{r-1}$ por (4.2).

- Si $x_j \neq \infty$, $1 \leq j \leq n$, entonces $d_j = 1$ y

$$\sum_{k=0}^{r-1} s_{ik} d_j z_j x_j^k = z_j \left(\sum_{k=0}^{r-1} s_{ik} x_j^k \right) = z_j (ax_j + b)^i = d_j z_j (ax_j + b)^i, \quad 0 \leq i \leq r-1.$$

- Si existe $j_0 \in \{1, \ldots, n\}$ tal que $x_{j_0} = \infty$, se tiene que si $j \neq j_0$, $1 \leq j \leq n$, estamos en el caso anterior y si $j = j_0$ entonces por un lado, la columna j-ésima de $A(ax_1 + b, \ldots, ax_n + b; z_1, \ldots, z_n)$ es $(0, \ldots, 0, z_j)^t$, y por otro lado, la columna j-ésima de $S_1 A(x_1, \ldots, x_n; d_1 z_1, \ldots, d_n z_n)$ es

$$(s_{0,r-1} d_j z_j, \ldots, s_{r-1,r-1} d_j z_j)^t = (0, \ldots, 0, a^{r-1} a^{-(r-1)} z_j)^t = (0, \ldots, 0, z_j)^t,$$

donde la primera igualdad se tiene por ser S_1 una matriz triangular inferior con $s_{r-1,r-1} = a^{r-1}$ y $d_j = a^{-(r-1)}$.

Se deduce así que

$$S_1 A(x_1, \ldots, x_n; d_1 z_1, \ldots, d_n z_n) = A(ax_1 + b, \ldots, ax_n + b; z_1, \ldots, z_n).$$

Además, como la matriz S_1 es triangular sin ceros en la diagonal, también es regular y por consiguiente, de la igualdad anterior se sigue que

$$SS_1^{-1} A(ax_1 + b, \ldots, ax_n + b; d_1^{-1} z_1, \ldots, d_n^{-1} z_n) = SA(x_1, \ldots, x_n; z_1, \ldots, z_n) = G',$$

por lo que

$$\left(SS_1^{-1}; ax_1 + b, \ldots, ax_n + b; d_1^{-1} z_1, \ldots, d_n^{-1} z_n\right) \tag{4.3}$$

es también solución de la ecuación (4.1).

De manera similar, considerando

$$S_2 = \begin{pmatrix} 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \end{pmatrix}, \quad \text{y} \quad d_j = \begin{cases} x_j^{-(r-1)} & \text{si} \quad x_j \notin \{0, \infty\} \\ \\ 1 & \text{si} \quad x_j \in \{0, \infty\} \end{cases}, \; 1 \leq j \leq n,$$

se tiene que:

- Si $x_j \notin \{0, \infty\}$, $1 \leq j \leq n$, entonces $d_j = x_j^{-(r-1)}$ y $d_j z_j x_j^i = z_j x_j^{i-(r-1)}$, para todo $i = 0, \ldots, r-1$, y todo $j = 1, \ldots, n$.
- Si existe $j_0 \in \{1, \ldots, n\}$ tal que $x_{j_0} = 0$, se tiene que si $j \neq j_0$, $1 \leq j \leq n$, estamos en el caso anterior y si $j = j_0$ entonces por un lado, la columna j-ésima de $A(x_1^{-1}, \ldots, x_n^{-1}; z_1, \ldots, z_n)$ es $(0, \ldots, 0, z_j)^t$, y por otro lado, la columna j-ésima de $S_2 A(x_1, \ldots, x_n; d_1 z_1, \ldots, d_n z_n)$ es

$$(0, \ldots, 0, d_j z_j)^t = (0, \ldots, 0, z_j)^t.$$

- Si existe $j_0 \in \{1, \ldots, n\}$ tal que $x_{j_0} = \infty$, se tiene que si $j \neq j_0$, $1 \leq j \leq n$, estamos en el primer caso y si $j = j_0$ entonces por un lado, la columna j-ésima de $A(x_1^{-1}, \ldots, x_n^{-1}; z_1, \ldots, z_n)$ es $(z_j, 0, \ldots, 0)^t$, y por otro lado, la columna j-ésima de $S_2 A(x_1, \ldots, x_n; d_1 z_1, \ldots, d_n z_n)$ es

$$(d_j z_j, 0, \ldots, 0)^t = (z_j, 0, \ldots, 0)^t.$$

Se deduce así que

$$S_2 A(x_1, \ldots, x_n; d_1 z_1, \ldots, d_n z_n) = A(x_1^{-1}, \ldots, x_n^{-1}; z_1, \ldots, z_n).$$

Además, por ser S_2 regular, de la igualdad anterior se deduce que

$$SS_2^{-1} A(x_1^{-1}, \ldots, x_n^{-1}; d_1^{-1} z_1, \ldots, d_n^{-1} z_n) = SA(x_1, \ldots, x_n; z_1, \ldots, z_n) = G',$$

por lo que

$$\left(SS_2^{-1}; x_1^{-1}, \ldots, x_n^{-1}; d_1^{-1}z_1, \ldots, d_n^{-1}z_n\right) \tag{4.4}$$

es también solución de (4.1).

Por otro lado, sabemos que la aplicación dada por $\phi(x) = \frac{ax+b}{cx+d}$, para todo $x \in \mathbb{F}_q$ con $ad-bc \neq 0$, es una composición de aplicaciones del tipo $x \longmapsto ax+b$ y $x \longmapsto \frac{1}{x}$.

Entonces, si $(S; x_1, \ldots, x_n; z_1, \ldots, z_n)$ es solución de (4.1), se deduce de (4.3) y (4.4) que existen $z'_1, \ldots, z'_n \in \mathbb{F}_q \setminus \{0\}$ y una matriz regular S_ϕ tales que

$$(SS_\phi^{-1}; \phi(x_1), \ldots, \phi(x_n); z'_1, \ldots, z'_n) \tag{4.5}$$

es solución de (4.1).

Lema 4.2.1.1. Sean $x_1, x_2, x_3 \in \mathbb{F}_q \setminus \{\infty\}$ distintos. Entonces existe una aplicación sobre $\mathbb{F}_q$ dada por $\phi(x) = \frac{ax+b}{cx+d}$ con $ad-bc \neq 0$, tal que $\phi(x_1) = 1$, $\phi(x_2) = 0$ y $\phi(x_3) = \infty$.

Demostración. Consideramos la aplicación dada por $\phi(x) = \frac{ax+b}{cx+d}$ tal que $ad-cb \neq 0$. En particular, tomando $a = 1, b = -x_2, c = \frac{x_1-x_2}{x_1-x_3}, d = -cx_3$, se tiene que

$$\phi(x) = \frac{x-x_2}{cx-cx_3}, \quad ad-cb = d+cx_2 = -cx_3+cx_2 = c(x_2-x_3) = \frac{(x_1-x_2)(x_2-x_3)}{x_1-x_3} \neq 0,$$

ya que por hipótesis x_1, x_2, x_3 son distintos.

Además, evaluando esta función en x_1, x_2 y x_3 se tiene que $\phi(x_1) = 1$, $\phi(x_2) = 0$ y $\phi(x_3) = \infty$. □

Del lema anterior y de (4.5) se deduce que existen $x_4, \ldots, x_n \in \mathbb{F}_q \setminus \{0, 1\}$, $z'_1, \ldots, z'_n \in \mathbb{F}_q \setminus \{0\}$ y una matriz regular S tales que

$$(S; 1, 0, \infty, x_4, \ldots, x_n; z'_1, \ldots, z'_n)$$

es solución de (4.1), y dado que es suficiente con conocer una solución, buscaremos una de esta forma, es decir, con $x_1 = 1$, $x_2 = 0$ y $x_3 = \infty$.

Teniendo en cuenta que $0 \leq i \leq r-1$ y $1 \leq j \leq n$, se puede reescribir la ecuación (4.1) como

$$G' = SA_1(x_1, \ldots, x_n)D = \left(g'_{ij}\right),$$

donde

$$S = (s_{ij}), \quad A_1(x_1, \ldots, x_n) = A(x_1, \ldots, x_n; 1, \ldots, 1), \quad D = \operatorname{diag}(z_1, \ldots, z_n), \tag{4.6}$$

con

$$SA_1(x_1, \ldots, x_n) = (a_{ij}), \quad a_{ij} = f_i(x_j), \quad f_i(x) = \sum_{k=0}^{r-1} s_{ik}x^k,$$

por lo que $g'_{ij} = z_j f_i(x_j)$.

Etapa 1:

Paso 1: Resolvemos los sistemas lineales homogéneos de $r-1$ ecuaciones y r incógnitas cada uno (por lo que tienen solución no trivial)

$$\sum_{i=0}^{r-1} c_{1i} g'_{ij} = 0, \qquad \forall j = 1, r+1, r+2, \ldots, 2(r-1),$$

$$\sum_{i=0}^{r-1} c_{2i} g'_{ij} = 0, \qquad \forall j = 2, r+1, r+2, \ldots, 2(r-1).$$

Paso 2: Sean

$$F_1(x) = \sum_{i=0}^{r-1} c_{1i} f_i(x) \ , \qquad \beta_{1j} = \sum_{i=0}^{r-1} c_{1i} g'_{ij} \ , \quad \forall j = 1, \ldots, n,$$

$$F_2(x) = \sum_{i=0}^{r-1} c_{2i} f_i(x) \ , \qquad \beta_{2j} = \sum_{i=0}^{r-1} c_{2i} g'_{ij} \ , \quad \forall j = 1, \ldots, n.$$

Para F_1 y β_{1j} se tiene que

$$\beta_{1j} = \sum_{i=0}^{r-1} c_{1i} z_j f_i(x_j) = z_j F_1(x_j) \text{ para } j = 1, \ldots, n.$$

Dado que $z_j \neq 0$, para todo $j = 1, \ldots, n$ y $\beta_{1j} = 0$ para $j = 1, r+1, r+2, \ldots, 2(r-1)$, se tiene que $x_1, x_{r+1}, x_{r+2} \ldots, x_{2(r-1)}$ son raíces finitas de $F_1(x)$ (nótese que $x_3 = \infty$ y $x_i \neq x_j$ si $i \neq j$). Además, $\deg(F_1) < r$ puesto que $\deg(f_i) < r$. Por lo tanto,

$$F_1(x) = a_1(x - x_1)(x - x_{r+1})(x - x_{r+2}) \cdots (x - x_{2(r-1)}),$$

y en particular, si $j \notin \{1, r+1, r+2, \ldots, 2(r-1)\}$, se tiene que $F_1(x_j) \neq 0$ luego $\beta_{1j} = z_j F_1(x_j) \neq 0$ y $\beta_{13} = z_3 F_1(z_3) = z_3 F_1(\infty) = a_1 z_3$.

Razonando de manera análoga se deduce que

$$\beta_{2j} = z_j F_2(x_j) \text{ para } j = 1, \ldots, n, \text{ y } F_2(x) = a_2(x-x_2)(x-x_{r+1})(x-x_{r+2}) \cdots (x-x_{2(r-1)}).$$

Paso 3: Calculamos $g'_j = \frac{\beta_{1j}}{\beta_{2j}}$, para $j = 3, \ldots, r, 2r-1, \ldots, n$, pues $\beta_{2j} \neq 0$ para esos valores de j. En particular, $g'_3 = \frac{\beta_{13}}{\beta_{23}} = \frac{a_1 z_1}{a_2 z_3} = \frac{a_1}{a_2}$, luego

$$g'_j = \frac{\beta_{1j}}{\beta_{2j}} = \frac{z_j F_1(x_j)}{z_j F_2(x_j)} = \frac{a_1(x_j - x_1)(x_j - x_{r+1}) \cdots (x_j - x_{2(r-1)})}{a_2(x_j - x_2)(x_j - x_{r+1}) \cdots (x_j - x_{2(r-1)})} = g'_3 \left(\frac{x_j - x_1}{x_j - x_2} \right).$$

Despejando x_j de la igualdad anterior y teniendo en cuenta que $x_1 = 1$ y $x_2 = 0$, se deduce que

$$x_j = \frac{g_3'}{g_3' - g_j'} \qquad \forall j = 4, \ldots, r, 2r-1, \ldots, n.$$

Paso 4: Resolvemos los sistemas lineales homogéneos de $r-1$ ecuaciones y r incógnitas cada uno (por lo que tienen solución no trivial)

$$\sum_{i=0}^{r-1} c_{3i} g_{ij}' = 0, \qquad \forall j = 1, 3, \ldots, r,$$

$$\sum_{i=0}^{r-1} c_{4i} g_{ij}' = 0, \qquad \forall j = 2, 3, \ldots, r.$$

Paso 5: Sean

$$F_3(x) = \sum_{i=0}^{r-1} c_{3i} f_i(x)\ , \qquad \beta_{3j} = \sum_{i=0}^{r-1} c_{3i} g_{ij}'\ , \quad \forall j = 1, \ldots, n,$$

$$F_4(x) = \sum_{i=0}^{r-1} c_{4i} f_i(x)\ , \qquad \beta_{4j} = \sum_{i=0}^{r-1} c_{4i} g_{ij}'\ , \quad \forall j = 1, \ldots, n.$$

Como $\beta_{3j} = 0$ y $z_j \neq 0$ para todo $j = 1, 3, \ldots, r$, se tiene que $F_3(x_j) = 0$, en particular, para $j = 3$, $0 = F_3(x_3) = F_3(\infty)$, por lo que el coeficiente de x^{r-1} en F_3 es cero, es decir, $\deg(F_3) < r-1$. Así, se deduce que

$$F_3(x) = a_3(x - x_1)(x - x_4)\cdots(x - x_r).$$

Razonando de manera análoga, se tiene que

$$F_4(x) = a_4(x - x_2)(x - x_4)\cdots(x - x_r).$$

Paso 6: Calculamos $\frac{\beta_{3j}}{\beta_{4j}}$ para todo $j = r+1, \ldots, n$, puesto que $\beta_{4j} \neq 0$ para esos valores de j,

$$\frac{\beta_{3j}}{\beta_{4j}} = \frac{z_j F_3(x_j)}{z_j F_4(x_j)} = \frac{a_3(x_j - x_1)}{a_4(x_j - x_2)}.$$

En particular, para $j = n$, teniendo en cuenta que $x_n = \frac{g_3'}{g_3' - g_n'}$, se tiene que

$$\frac{\beta_{3n}}{\beta_{4n}} = \frac{a_3(x_n - x_1)}{a_4(x_n - x_2)} = \frac{a_3 g_n'}{a_4 g_3'},$$

luego

$$\frac{a_3}{a_4} = \frac{g_3' \beta_{3n}}{g_n' \beta_{4n}} \quad \text{y} \quad \frac{\beta_{3j}}{\beta_{4j}} = \frac{g_3' \beta_{3n}(x_j - x_1)}{g_n' \beta_{4n}(x_j - x_2)}.$$

Sea $g'_j = \frac{\beta_{4n}\beta_{3j}}{\beta_{3n}\beta_{4j}} g'_n$ para $j = r+1, \ldots, 2(r-1)$. Entonces, para esos valores de j se tiene que

$$g'_j = g'_3 \frac{x_j - x_1}{x_j - x_2} \quad \text{y} \quad x_j = \frac{g'_3}{g'_3 - g'_j}.$$

Así, $x_j = \frac{g'_3}{g'_3 - g'_j}$ para todo $j = 1, \ldots, n$ donde $g'_j = g'_3 \frac{x_j - x_1}{x_j - x_2}$ y $g'_3 = \frac{a_1}{a_2}$.

En el algoritmo 13 se muestra la primera etapa del ataque, donde se ha calculado el parámetro $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$.

Nota. El último paso del algoritmo 13, es una solución de (4.1), donde todos los x_j, $j = 1, \ldots, n$ son finitos.

La complejidad total del algoritmo 13 es cúbica, $O((r-1)^3)$, ya que resolver cada uno de los cuatro sistemas lineales de $r-1$ ecuaciones tiene complejidad $O((r-1)^3)$, las líneas 2 y 4 del algoritmo tienen complejidad $O(n(r-1))$ y $O((r-1)^2)$, respectivamente, y las dos últimas líneas del código tienen cada una complejidad $O(n)$.

Algorithm 13: Cálculo de $x_1, \ldots, x_n$

Input : La clave pública $G' \in \mathfrak{M}_{k\times n}(\mathbb{F}_q)$.

Output: El parámetro $\mathbf{x} = (x_1, \ldots, x_n)$.

1 Encontrar $c_{1i}, c_{2i} \in \mathbb{F}_q$, $0 \leq i \leq r-1$ tales que son solución no trivial de los sistemas lineales

$$\sum_{i=0}^{r-1} c_{1i} g'_{ij} = 0, \qquad \forall j = 1, r+1, r+2, \ldots, 2(r-1),$$

$$\sum_{i=0}^{r-1} c_{2i} g'_{ij} = 0, \qquad \forall j = 2, r+1, r+2, \ldots, 2(r-1).$$

2 Calcular

$$\beta_{1j} = \sum_{i=0}^{r-1} c_{1i} g'_{ij} \ , \quad \beta_{2j} = \sum_{i=0}^{r-1} c_{2i} g'_{ij} \ , \quad \forall j = 3, \ldots, r, 2r-1, \ldots, n,$$

y encontrar $g'_j = \dfrac{\beta_{1j}}{\beta_{2j}}$ para esos valores de j.

3 Encontrar $c_{3i}, c_{4i} \in \mathbb{F}_q$, $0 \leq i \leq r-1$ tales que son solución no trivial de los sistemas lineales

$$\sum_{i=0}^{r-1} c_{3i} g'_{ij} = 0, \qquad \forall j = 1, 3, \ldots, r,$$

$$\sum_{i=0}^{r-1} c_{4i} g'_{ij} = 0, \qquad \forall j = 2, 3, \ldots, r.$$

4 Calcular

$$\beta_{3j} = \sum_{i=0}^{r-1} c_{3i} g'_{ij} \ , \quad \beta_{4j} = \sum_{i=0}^{r-1} c_{4i} g'_{ij} \ , \quad \forall j = r+1, \ldots, 2(r-1), n,$$

y encontrar $g'_j = g'_n \dfrac{\beta_{4n}\beta_{3j}}{\beta_{3n}\beta_{4j}}$ para $j = r+1, \ldots, 2(r-1)$.

5 Poner $x_1 = 1$, $x_2 = 0$, $x_3 = \infty$ y $x_j = \dfrac{g'_3}{g'_3 - g'_j}$ para $j = 4, \ldots, n$.

6 Elegir al azar $a \in \mathbb{F}_q \setminus \{0, x_1, \ldots, x_n\}$ y reescribir $x_j = \dfrac{1}{a - x_j}$, $j = 1, \ldots, n$.

7 **return** $\mathbf{x} = (x_1, \ldots, x_n)$

Etapa 2: Una vez hemos encontrado $x_1, \ldots, x_n$, vamos a calcular $z_1, \ldots, z_n \in \mathbb{F}_q \setminus \{0\}$ y la matriz regular S.

Podemos suponer sin pérdida de generalidad que $z_1 = 1$, ya que si se fija un $a \in \mathbb{F}_q \setminus \{0\}$, el producto matricial SA_1D es el mismo que si multiplicamos por a cada elemento de D y por a^{-1} cada entrada de S, donde A_1 y D son las matrices dadas en (4.6).

Paso 1: Resolvemos el sistema lineal homogéneo de r ecuaciones y $r+1$ incógnitas (por lo que tiene solución no trivial)

$$\sum_{j=1}^{r+1} c_j g'_{ij} = 0, \qquad \forall i = 0, \ldots, r-1. \tag{4.7}$$

Nótese que todos los $c_j \in \mathbb{F}_q$, $j = 1, \ldots, r+1$ son no nulos. Si uno de ellos fuera cero, entonces la matriz G' tendría r columnas linealmente dependientes. Por otro lado, G' es una matriz de control de un código Reed-Solomon generalizado de dimensión $n-r$, y por ser estos códigos MDS se tiene que $d = n-(n-r)+1 = r+1$. Además, de la proposición 1.2.2.3 se tiene que d es el menor número de columnas linealmente dependientes de G', es decir $d = r$, pero esto es absurdo pues $r \neq r+1$.

Paso 2: Reescribimos la ecuación (4.7), teniendo en cuenta que $g'_{ij} = z_j f_i(x_j)$, como

$$\sum_{j=1}^{r+1} c_j z_j f_i(x_j) = 0, \qquad \forall i = 0, \ldots, r-1,$$

o matricialmente,

$$AC\mathbf{z}' = \mathbf{0},$$

donde

$$A = (a_{ij}) \ , \quad a_{ij} = f_i(x_j) \ , \quad 0 \leq i \leq r-1 \ , \ 1 \leq j \leq r+1 \ ,$$
$$C = \operatorname{diag}(c_1, \ldots, c_{r+1}) \ , \quad \mathbf{z}' = (z_1, \ldots, z_{r+1})^t.$$

Dado que $A = SA_1(x_1, \ldots, x_{r+1})$, la igualdad anterior se puede escribir como

$$SA_1(x_1, \ldots, x_{r+1})C\mathbf{z}' = \mathbf{0},$$

y multiplicando por S^{-1} se tiene que

$$A_1(x_1, \ldots, x_{r+1})C\mathbf{z}' = \mathbf{0},$$

es decir

$$\sum_{j=1}^{r+1} c_j z_j x_j^i = 0 \ , \qquad \forall i = 0, \ldots, r-1,$$

que es un sistema lineal homogéneo de r ecuaciones y r incógnitas $z_2, \ldots, z_{r+1}$, puesto que $z_1 = 1$ y los valores de c_j y x_j son conocidos para $j = 1, \ldots, r+1$.

Tal sistema tiene una única solución no trivial ya que, por ser $c_j \neq 0$ para $j = 1, \ldots, r+1$, y $A_1(x_2, \ldots, x_{r+1})$ una matriz cuadrada de tipo Vandermonde, su determinante es $c_2 \cdots c_{r+1} \det(A_1(x_2, \ldots, x_{r+1})) \neq 0$.

Resolviendo el sistema obtenemos los valores de $z_1, z_2, \ldots, z_{r+1} \in \mathbb{F}_q \setminus \{0\}$.

Paso 3: Como $G' = SA_1(x_1, \ldots, x_n)D$, se tiene que

$$g'_{ij} = z_j \sum_{k=0}^{r-1} s_{ik} x_j^k \ , \quad \forall i = 0, \ldots, r-1 \ , \ j = 1, \ldots, n.$$

Entonces, para cada $i \in \{0, \ldots, r-1\}$, como conocemos los valores no nulos de $z_1, \ldots, z_r$, se tiene un sistema lineal de r ecuaciones con r incógnitas

$$\sum_{k=0}^{r-1} s_{ik} x_j^k = g'_{ij} z_j^{-1} \ , \quad \forall j = 1, \ldots, r,$$

que tiene una única solución, por ser la matriz del sistema cuadrada y de tipo Vandermonde.

Resolviendo cada uno de los sistemas correspondientes para cada $i = 0, \ldots, r-1$, recuperamos la matriz regular S.

Paso 4: Multiplicando la ecuación (4.1) por $S^{-1} = \left(s'_{ij}\right)$, se tiene que

$$A(x_1, \ldots, x_n; z_1, \ldots, z_n) = S^{-1} G',$$

y como la primera fila de $A(x_1, \ldots, x_n; z_1, \ldots, z_n)$ es $(z_1, \ldots, z_n)$, se deduce que

$$z_j = \sum_{i=0}^{r-1} s'_{0i} g'_{ij} \ , \quad \forall j = 1, \ldots, n,$$

y en particular para $j = r+2, \ldots, n$, donde los s'_{0i}, $i = 0, \ldots, r-1$ se calculan resolviendo el sistema

$$\sum_{i=0}^{r-1} s'_{0i} s_{i0} = 1 \ , \quad \sum_{i=0}^{r-1} s'_{0i} s_{ij} = 0 \ , \quad \forall j = 1, \ldots, r-1.$$

Observemos que aunque no es necesario calcular todos los elementos de S^{-1} para obtener los valores de $z_{r+2}, \ldots, z_n$, habrá que conocerlos para después descodificar el mensaje $\mathbf{c}$.

Algorithm 14: Cálculo de $z_1, \ldots, z_n$ y S

Input : La clave pública $G' \in \mathfrak{M}_{k\times n}(\mathbb{F}_q)$ y el parámetro $\mathbf{x} = (x_1, \ldots, x_n)$.

Output: El parámetro $\mathbf{z} = (z_1, \ldots, z_n)$ y la matriz regular S.

1 Encontrar $c_1, \ldots c_{r+1} \in \mathbb{F}_q$, tales que son solución no trivial del sistema lineal

$$\sum_{j=1}^{r+1} c_j g'_{ij} = 0, \qquad \forall i = 0, \ldots, r-1.$$

2 Poner $z_1 = 1$ y calcular $z_2, \ldots, z_{r+1} \in \mathbb{F}_q$ tales que sean solución del sistema lineal

$$\sum_{j=1}^{r+1} c_j z_j x_j^i = 0, \qquad \forall i = 0, \ldots, r-1.$$

3 Para cada $i = 0, \ldots, r-1$, encontrar las soluciones $s_{i0}, \ldots, s_{i(r-1)} \in \mathbb{F}_q$, del sistema lineal

$$\sum_{k=0}^{r-1} s_{ik} x_j^i = z_j^{-1} g'_{ij}, \qquad \forall j = 1, \ldots, r,$$

y construir $S = (s_{ij})$.

4 Encontrar la matriz $S^{-1} = \left(s'_{ij}\right)$ y calcular

$$z_j = \sum_{i=0}^{r-1} s'_{0i} g'_{ij}, \qquad \forall j = r+2, \ldots, n.$$

5 **return** $\mathbf{z} = (z_1, \ldots, z_n)$ y S.

En el algoritmo 14 se muestra la segunda etapa del ataque, donde se han calculado el parámetro $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}_q^n$, con $z_j \neq 0$ para todo $j = 1, \ldots, n$, y la matriz regular S.

La complejidad total del algoritmo 14 es polinómica, $O(r^4 + (n-r)r)$. Esto se deduce de que la complejidad de resolver cada uno de los sistemas lineales de las líneas 1 y 2 es cúbica $O((r-1)^3)$, la línea 3 tiene complejidad $O((r-1)^4)$, y la última línea del algoritmo tiene complejidad $O((r-1)^4 + (n-r-1)r) = O(r^4 + (n-r)r)$ puesto que calcular S^{-1} tiene complejidad $O((r-1)^4)$ y calcular cada uno de los $n-(r+1)$ z_j tiene complejidad $O(r)$.

Comparando las complejidades de ambas etapas, se concluye que el ataque propuesto por Sidelnikov y Shestakov tiene complejidad polinómica $O(r^4 + (n-r)r)$, por lo que es un algoritmo eficiente que puede romper el criptosistema de McEliece si se han utilizado códigos Reed-Solomon generalizados para su definición.

Conclusiones

Los criptosistemas de clave pública más utilizados hoy en día son el criptosistema RSA, cuya seguridad reside en factorizar un entero en números primos, y ElGamal, que está basado en el problema del logaritmo discreto (PLD). Además, como vimos al final de la sección 3.2, el tamaño de las claves de ambos criptosistemas es lineal en el tamaño del mensaje. El objetivo principal del trabajo era desarrollar el criptosistema de McEliece, otro criptosistema de clave pública, aunque no se utiliza en la práctica debido a que el tamaño de sus claves es polinómico en el tamaño del mensaje. Entonces, ¿por qué este criptosistema tiene tanto interés?

En 1999, P.W. Shor desarrolló en [7] cómo se puede factorizar eficientemente un entero en números primos utilizando el muestreo de Fourier cuántico sobre el grupo cíclico $\mathbb{Z}_n$. También, en el mismo artículo, fue capaz de resolver en tiempo polinómico el PLD utilizando las técnicas mencionadas anteriormente. Así, consiguió romper tanto el RSA como los criptosistemas basados en el PLD utilizando ataques con ordenadores cuánticos.

Actualmente, este tipo de ordenadores solo existen en el marco teórico, pero cuando se conviertan en algo tangible, estos criptosistemas de clave pública ya no serán seguros y quedarán obsoletos.

No obstante, el criptosistema de McEliece, también de clave pública, es capaz de resistir tales ataques cuánticos, como se puede ver en [1], siempre que el código que lo defina tenga ciertas propiedades algebraicas, como por ejemplo los códigos Goppa. Esta propiedad dota al criptosistema de McEliece de una ventaja significativa frente a otros criptosistemas de clave pública y además, esto le convierte en uno de los candidatos principales a criptosistema de clave pública seguro frente a ataques cuánticos. Por ello, aunque el tamaño de las claves del criptosistema de McEliece sea muy grande en comparación con el del RSA y ElGamal, el coste merecerá la pena, siempre que la seguridad esté garantizada.

Destacar, que aunque el criptosistema de McEliece resista ataques cuánticos para el código que lo defina, también hay que tener en cuenta que este ha de ser seguro frente a ataques genéricos y ataques estructurales, pues sino podrá romperse con técnicas clásicas. Ha día de hoy, aunque el criptosistema de McEliece resista

ataques genéricos para el código a partir del cual se ha construido, existen ataques estructurales que lo harían vulnerable, a no ser que se considere un código Goppa. Como hemos visto, por ejemplo, el criptosistema de McEliece con códigos Reed-Solomon generalizados admiten ataques estructurales en tiempo polinómico. Otros códigos con estructura muy fuerte, como los cíclicos o los Reed-Muller, también admiten ataques estructurales.

Se concluye por lo tanto que este criptosistema es seguro si se define a partir de códigos Goppa, pues estos son capaces de resistir tanto a los ataques genéricos y a los ataques estructurales como a los ataques con ordenadores cuánticos.

Bibliografía

[1] Hang Dinh, Cristopher Moore, and Alexander Russell. Mceliece and niederreiter cryptosystems that resist quantum fourier sampling attacks. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 761–779. Springer, 2011.

[2] Tom Høholdt and Jørn Justesen. *A course in error-correcting codes*. 2017.

[3] Pil Joong Lee and Ernest F Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Advances in Cryptology—EUROCRYPT'88: Workshop on the Theory and Application of Cryptographic Techniques Davos, Switzerland, May 25–27, 1988 Proceedings 7*, pages 275–280. Springer, 1988.

[4] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[5] RJ McEliece. A public-key cryptosystem based on algebraic coding theory. dsn progress report 42-44, jet propulsion lab. 1978.

[6] Ruud Pellikaan, Xin-Wen Wu, Stanislav Bulygin, and Relinde Jurrius. *Codes, cryptology and curves with computer algebra*. Cambridge University Press, 2017.

[7] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[8] Vladimir Michilovich Sidelnikov and Sergey O Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. 1992.

[9] C Munuera-J Tena. Codificación de la información, ed. *Universidad de Valladolid*, 1997.

[10] HCA Van Tilborg, RJ Mceliece, and ER Berlekamp. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 1978.

Índice de figuras