UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

# CRIPTOANÁLISIS DEL CRIPTOSISTEMA DE MCELIECE CLÁSICO MEDIANTE ALGORITMOS GENÉTICOS


**Autor**

PAULA VILLANUEVA NÚÑEZ

**Director**

GABRIEL NAVARRO GARULO


Facultad de Ciencias

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación


FACULTAD DE CIENCIAS

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN


*Granada, a 23 de febrero de 2022*

# ÍNDICE GENERAL

## RESUMEN

Añadir resumen

PALABRAS CLAVE:

SUMMARY

Añadir resumen en inglés

KEYWORDS:

# INTRODUCCIÓN Y OBJETIVOS

Añadir introducción

# 1

# PRELIMINARES

En este capítulo se desarrollarán las herramientas necesarias para poder afrontar el criptosistema de McEliece que precisa este trabajo. Se abordarán conceptos relacionados con el álgebra lineal, anillos, cuerpos finitos, polinomios, algoritmos genéticos, etc.

## 1.1 ANILLOS

En esta sección introduciremos el concepto de anillo para poder definir el concepto de cuerpo.

**Definición 1.** Un *anillo* $(A, +, \cdot)$ es un conjunto $A$ junto con dos operaciones binarias $A \times A \to A$ denotadas por la suma (denotada por $+$) y producto (denotado por $\cdot$) que verifican los siguientes axiomas:

- Propiedad asociativa de la suma:

$$a + (b + c) = (a + b) + c \qquad \forall a, b, c \in A$$

- Existencia del elemento neutro para la suma:

$$0 + a = a = a + 0 \qquad \forall a \in A$$

- Existencia del elemento inverso para la suma:

$$\forall a \in A \ \ \exists - a \in A \quad a + (-a) = 0 = (-a) + a$$

- Propiedad conmutativa de la suma:

$$a + b = b + a \qquad \forall a, b \in A$$

- Propiedad asociativa del producto:

$$a(bc) = (ab)c \qquad \forall a, b, c \in A$$

- Propiedad distributiva del producto:

$$a(b + c) = ab + ac, \quad (b + c)a = ba + ca \qquad \forall a, b, c \in A$$

- Existencia del elemento neutro para el producto:

$$1a = a = a1 \qquad \forall a \in A$$

Un anillo de llama *conmutativo o abeliano* si se verifica la propiedad conmutativa del producto

$$ab = ba \qquad \forall a, b \in A$$

Añadir algo de ideales.

## 1.2 CUERPOS FINITOS

Con estos conceptos previos podemos ya definir el de cuerpo.

**Definición 2.** Un *cuerpo* $(A, +, \cdot)$ es un anillo conmutativo no trivial en el que todo elemento no nulo tiene un inverso multiplicativo. Se dice que un cuerpo es *finito* si tiene un número finito de elementos.

En los códigos lineales son comunes los siguientes cuerpos: *cuerpo binario* con dos elementos, *cuerpo ternario* con tres elementos y *cuerpo cuaternario* con cuatro elementos.

Diremos que la *característica* de un cuerpo es el número de elementos que tiene.

Todos los cuerpos finitos tienen un número de elementos $q = p^n$, para algún número primo $p$ y algún entero positivo $n$. Denotaremos por $\mathbb{F}_q$ a los cuerpos finitos con característica $q$.

Hay que definir las clases ciclotómicas

## 1.3 POLINOMIOS

En esta sección vamos a introducir el concepto de polinomio junto con sus operaciones.

**Definición 3.** Sea $A$ un anillo conmutativo. El *conjunto de polinomios* en la indeterminada $X$ con coeficientes en $A$ es el conjunto de todas las sumas formales finitas

$$f = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$$

Este conjunto de representa por $A[X]$.

En el conjunto de polinomios definimos una suma y un producto.

Sean $f = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$ y $g = b_m X^m + b_{m-1} X^{m-1} + \cdots + b_1 X + b_0$ dos polinomios. Supongamos que $m \leq n$, tomando $b_i = 0$ para todo $n \geq i > m$, definimos las operaciones de suma y producto de polinomios

$$f + g = (a_n + b_n) X^n + \cdots + (a_1 + b_1) X + (a_0 + b_0)$$

$$f \cdot g = a_n b_m X^{n+m} + (a_n b_{m+1} + a_{n-1} b_m) X^{n+m-1} + \cdots + (a_1 b_0 + a_0 b_1) X + a_0 b_0$$

De esta forma, diremos que el conjunto $A[X]$ con las operaciones anteriores es un *anillo de polinomios en $X$ con coeficientes en $A$*.

**Definición 4.** Para un polinomio $f = a_n + a_{n-1}X^{n-1} + \cdots + a_1X + a_0 \neq 0$ el mayor índice $n$ tal que $a_n \neq 0$ se llama *grado de* $f$ y se representa por $gr(f)$. Si $f = 0$ definimos $gr(f) = -\infty$.

Llamaremos *término (de grado i)* a cada uno de los sumandos $a_iX^i$ del polinomio $f$. El *término líder* es el término no nulo de mayor grado. El coeficiente $a_n \neq 0$ del término líder se llama *coeficiente líder* y el término de grado cero $a_0$ se llama *término constante*.

A continuación tenemos algunas propiedades de los polinomios.

**Proposición 1.** *Sea $A$ un anillo conmutativo y sean $f, g \in A[X]$ dos polinomios, tenemos que*

$$gr(f+g) \leq \text{máx}\,(gr(f), gr(g)),$$

$$gr(f \cdot g) \leq gr(f) + gr(g)$$

*Si $gr(f) \neq gr(g)$, se verifica*

$$gr(f+g) = \text{máx}\,(gr(f), gr(g))$$

*Si $A$ es un dominio de integridad, entonces*

$$gr(f \cdot g) = gr(f) + gr(g)$$

Añadir más propiedades

## 1.4 ALGORITMOS GENÉTICOS

Añadir introducción a la sección: qué es y para qué sirve (tratar problemas que no son P, o que no se sabe que sean P). Didáctica!!!!

Los *algoritmos genéticos* son algoritmos de optimización, búsqueda y aprendizaje inspirados en los procesos de evolución natural y evolución genética.

En general, los algoritmos genéticos siguen el siguiente procedimiento (explicarlo mejor).

```
t = 0
inicializar la poblacion P(t)
evaluar la poblacion P(t)
Mientras (no se cumpla la condicion de parada) hacer
    t = t + 1
    seleccionar P' desde P(t-1)
    recombinar P'
    mutar P'
    reemplazar P(t) a partir de P(t-1) y P'
    evaluar P(t)
```

Existen dos modelos de algoritmos genéticos, el modelo generacional y el modelo estacionario.

En el modelo generacional, durante cada iteración se crea una población completa con nuevos individuos. Así, la nueva población reemplaza directamente a la antigua.

En el modelo estacionario, durante cada iteración se escogen dos padres de la población y se les aplican los operadores genéticos. De este modo, los descendientes reemplazan a los cromosomas de la población anterior. Este modelo es elitista y produce una convergencia rápida cuando se reemplazan los peores cromosomas de la población.

Añadir más cosas

## 1.5 CLASES DE COMPLEJIDAD

Explicar que es un problema NP-completo y demás.

# 2

# INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS LINEALES

Hay que reescribir este párrafo

El inicio de la teoría de códigos surgió a partir de la publicación de Claude Shannon sobre "Una teoría matemática sobre la comunicación" en 1948 [1]. En este artículo, Shannon explica que es posible transmitir mensajes fiables en un canal de comunicación que puede corromper la información enviada a través de él siempre y cuando no se supere la capacidad de dicho canal.

Con la teoría de códigos, podemos codificar datos antes de transmitirlos de tal forma que los datos alterados puedan ser decodificados al grado de precisión especificado. Así, el principal problema es determinar el mensaje que fue enviado a partir del recibido. El Teorema de Shannon nos garantiza que el mensaje recibido coincidirá con el que fue enviado un cierto porcentaje de las veces. Esto hace que el objetivo de la teoría de códigos sea crear códigos que cumplan las condiciones de este teorema.

En esta sección introduciremos los conceptos y resultados fundamentales sobre la teoría de códigos lineales. El desarrollo de este capítulo se ha basado en [2], [3], [4] y [5].

## 2.1 INTRODUCCIÓN

Supongamos que queremos enviar un mensaje, por lo que habrá un emisor y un receptor que se comunican en una dirección. Este mensaje es enviado por un *canal de comunicación*, cuyas características dependen de la naturaleza del mensaje a ser enviado. En general, hay que hacer una *traducción* entre el mensaje original (o *palabra fuente*) $x$ y el tipo de mensaje $c$ que el canal está capacitado para enviar (*palabras código*). Este proceso se llama *codificación*. Una vez codificado el mensaje, lo enviamos a través del canal, y nuestro intermediario (el receptor) recibe un mensaje codificado (*palabra recibida*) posiblemente erróneo, ya que en todo proceso de comunicación hay ruido e interferencias. Una vez recibido, empieza el proceso llamado *corrección de errores*, que consiste en recuperar el mensaje original corrigiendo los errores que se hubieran producido. El mensaje recibido $c'$ es traducido nuevamente a términos originales $x'$, es decir, es *decodificado*. La siguiente figura representa un esquema de este proceso.

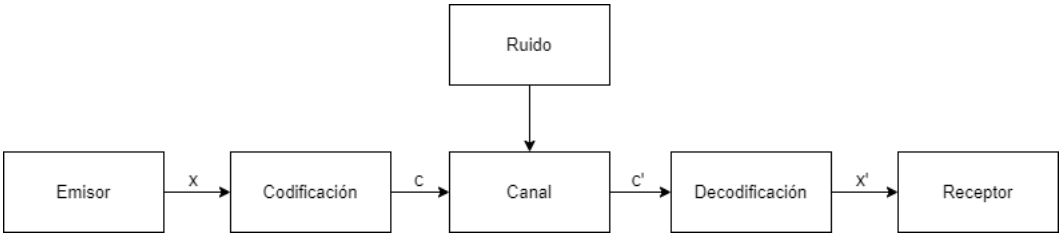


Figura 1: Esquema del modelo de comunicación

Las flechas indican que la comunicación es en un solo sentido.

En general, $x' \neq x$ y es deseable que este error sea detectado (lo cual permite pedir una retransmisión del mensaje) y en lo posible corregido.

La *Teoría de Códigos Autocorrectores* se ocupa del segundo y cuarto pasos del esquema anterior, es decir, de la codificación y decodificación de mensajes, junto con el problema de detectar y corregir errores. A veces no es posible pedir retransmisión de mensajes y es por eso que los códigos autocorrectores son tan útiles y necesarios.

La calidad de un código con mensajes de longitud $k$ y palabras código de longitud $n$ vendrá dada por las siguientes características.

- El cociente $\frac{k}{n}$, el *ratio de información* del código, que mide el esfuerzo necesario para transmitir un mensaje codificado.
- La *distancia mínima relativa* $\frac{d}{n}$ que es aproximadamente el doble de la proporción de errores que se pueden corregir en cada mensaje codificado.
- La *complejidad* de los procedimientos de codificar y decodificar.

De esta forma, uno de los objetivos centrales de la teoría de códigos autocorrectores es construir códigos que sean de calidad. Esto es, códigos que permitan codificar muchos mensajes, que se puedan trasmitir rápida y eficientemente, que detecten y corrijan simultáneamente la mayor cantidad de errores posibles y que haya algoritmos de decodificación fáciles y efectivos. Por lo que habrá que encontrar un balance entre estas distintas metas, pues suelen ser contradictorias entre sí.

## 2.2 CÓDIGOS LINEALES

Consideramos $\mathbb{F}_q^n$, que denota al espacio vectorial de las n-tuplas sobre el cuerpo finito $\mathbb{F}_q$. Generalmente los vectores $(a_1, ..., a_n)$ de $\mathbb{F}_q^n$ se denotarán por $a_1 \cdots a_n$.

**Definición 5.** Un $(n, M)$ *código* $\mathcal{C}$ sobre $\mathbb{F}_q$ es un subconjunto de $\mathbb{F}_q^n$ de tamaño $M$. A los elementos de $\mathcal{C}$ los llamaremos *palabras código*.

*Ejemplo* 1.

- Un código sobre $\mathbb{F}_2$ se llama *código binario* y un ejemplo es $\mathcal{C} = \{00, 01, 10, 11\}$.
- Un código sobre $\mathbb{F}_3$ se llama *código ternario* y un ejemplo es $\mathcal{C} = \{21, 02, 10, 20\}$.

Con el fin de aportar más utilidad a los códigos, imponemos linealidad. Así, si $\mathcal{C}$ un subespacio k-dimensional de $\mathbb{F}_q^n$, entonces decimos que $\mathcal{C}$ es un $[n,k]$ *código lineal* sobre $\mathbb{F}_q$. De esta forma, los códigos lineales tendrán $q^k$ palabras código. Estos se pueden presentar con una matriz generadora o con una matriz de paridad.

**Definición 6.** Una *matriz generadora* para un $[n,k]$ código $\mathcal{C}$ es una matriz $k \times n$ donde sus filas forman una base de $\mathcal{C}$.

**Definición 7.** Para cada conjunto de $k$ columnas independientes de una matriz generadora $G$, se dice que el conjunto de coordenadas correspondiente conforman un *conjunto de información* de $\mathcal{C}$. Las $r = n - k$ restantes coordenadas se denominan *conjunto de redundancia* y el número $r$ es la *redundancia* de $\mathcal{C}$.

En general, la matriz generadora no es única. Sin embargo, si las $k$ primeras coordenadas conforman un conjunto de información, entonces el código tiene una única matriz generadora de la forma $(I_k|A)$, donde $I_k$ denota a la matriz identidad $k \times k$. Esta matriz se dice que está en *forma estándar*.

Como un código lineal es un subespacio de un espacio vectorial, es el núcleo de alguna transformación lineal.

**Definición 8.** Una *matriz de paridad* $H$ de dimensión $(n-k) \times n$ de un $[n,k]$ código $\mathcal{C}$ se define como

$$C = \left\{ \mathbf{x} \in \mathbb{F}_q^n : H\mathbf{x}^T = 0 \right\}.$$

Al igual que con la matriz generadora, la matriz de paridad no es única. Con el siguiente resultado podremos obtener una matriz de paridad cuando $\mathcal{C}$ tiene una matriz generadora en forma estándar.

**Teorema 1.** *Si $G = (I_k|A)$ es una matriz generadora para el $[n,k]$ código $\mathcal{C}$ en forma estándar, entonces $H = \left(-A^T|I_{n-k}\right)$ es una matriz de paridad de $\mathcal{C}$.*

**Demostración.** Como $HG^T = -A^T + A^T = 0$, se tiene que $\mathcal{C}$ está contenido en el núcleo de la transformación lineal $x \mapsto Hx^T$. Esta transformación lineal tiene un núcleo de dimensión $k$, pues $H$ tiene rango $n-k$, que coincide con la dimensión de $\mathcal{C}$. □

*Ejemplo* 2. Sea la matriz $G = (I_4|A)$, donde

$$G = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

es una matriz generadora en forma estándar para un $[7,4]$ código binario que denotaremos por $\mathcal{H}_3$. Por el Teorema 1, una matriz de paridad de $\mathcal{H}_3$ es

$$H = \left( \begin{array}{c|c} -A^T & I_{7-4} \end{array} \right) = \left( \begin{array}{c|c} -A^T & I_3 \end{array} \right) = \left( \begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right)$$

## 2.3 CÓDIGO DUAL

Sabemos que $\mathcal{C}$ es un subespacio de un espacio vectorial, por lo que podemos calcular un subespacio ortogonal a dicho subespacio y así obtener lo que se denomina *espacio dual u ortogonal* de $\mathcal{C}$, denotado por $\mathcal{C}^\perp$. Se define este concepto con la operación del producto escalar como sigue.

**Definición 9.** El *espacio dual* de $\mathcal{C}$ viene dado por

$$\mathcal{C}^\perp = \left\{ \mathbf{x} \in \mathbb{F}_q^n \ : \ \mathbf{x} \cdot \mathbf{c} = 0 \quad \forall \mathbf{c} \in \mathcal{C} \right\}$$

Se observa que $\mathcal{C}^\perp$ es un $[n, n-k]$ código.

El siguiente resultado nos muestra cómo obtener las matrices generadora y de paridad de $\mathcal{C}^\perp$ a partir de las de $\mathcal{C}$.

**Proposición 2.** *Si tenemos una matriz generadora $G$ y una matriz de paridad $H$ de un código $\mathcal{C}$, entonces $H$ y $G$ son matrices generadoras y de paridad, respectivamente, de $\mathcal{C}^\perp$.*

Diremos que un código $\mathcal{C}$ es *auto-ortogonal* si $\mathcal{C} \subseteq \mathcal{C}^\perp$ y *auto-dual* cuando $\mathcal{C} = \mathcal{C}^\perp$.

*Ejemplo* 3. Una matriz generadora para el $[7, 4]$ código de Hamming $\mathcal{H}_3$ se presenta en el Ejemplo 2. Sea $\mathcal{H}'_3$ el código de longitud 8 y dimensión 4 obtenido de $\mathcal{H}_3$ añadiendo una coordenada de verificación de paridad general a cada vector de G y por lo tanto a cada palabra código de $\mathcal{H}_3$. Entonces

$$G' = \left( \begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array} \right)$$

es una matriz generadora para $\mathcal{H}'_3$. Además, veamos que $\mathcal{H}'_3$ es un código auto-dual.

Tenemos que $G' = (I_4 | A')$, donde

$$A' = \left( \begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} \right).$$

Como $A'(A')^T = I_4$, entonces $\mathcal{H}'_3$ es auto-dual.

## 2.4 PESOS Y DISTANCIAS

Es importante saber lo que difieren las palabras código. En este apartado estudiaremos esta idea y cómo puede influir a la teoría de códigos.

**Definición 10.** La *distancia de Hamming* $\mathrm{d}(\mathbf{x}, \mathbf{y})$ entre dos vectores $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ se define como el número de coordenadas en las que $\mathbf{x}$ e $\mathbf{y}$ difieren.

*Ejemplo* 4. Sean $\mathbf{x} = 012$, $\mathbf{y} = 210$ $\mathbf{x}, \mathbf{y} \in \mathbb{F}_3^4$. Entonces la distancia de Hamming entre los dos vectores es $d(\mathbf{x}, \mathbf{y}) = 2$.

**Teorema 2.** *La función distancia* $\mathrm{d}(\mathbf{x}, \mathbf{y})$ *satisface las siguientes propiedades.*

1. *No negatividad:* $\mathrm{d}(\mathbf{x}, \mathbf{y}) \geq 0 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$.
2. $\mathrm{d}(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$.
3. *Simetría:* $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{d}(\mathbf{y}, \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$.
4. *Desigualdad triangular:* $\mathrm{d}(\mathbf{x}, \mathbf{z}) \leq \mathrm{d}(\mathbf{x}, \mathbf{y}) + \mathrm{d}(\mathbf{y}, \mathbf{z}) \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$

**Demostración.** Las tres primeras afirmaciones se obtienen directamente a partir de la definición. La cuarta propiedad se obtiene a partir de la no negatividad. Esto es, sean $x, y, z \in \mathbb{F}_q^n$ distingamos dos casos. Si $x \neq z$ tenemos que $y \neq x$ o $y \neq z$, entonces por la no negatividad se cumple la afirmación. En el caso en el que $x = z$, tendríamos que $d(x, z) = 0$ y también se da la afirmación. □

Diremos que la *distancia mínima* de un código $\mathcal{C}$ es la distancia más pequeña entre las distintas palabras código. Esta medida es fundamental a la hora de determinar la capacidad de corregir errores de $\mathcal{C}$.

*Ejemplo* 5. Sea $\mathcal{C} = 010101, 212121, 111000$ un código ternario. Entonces

$$d(010101, 212121) = 3, \qquad d(010101, 111000) = 4, \qquad d(212121, 111000) = 5.$$

Por lo que la distancia mínima del código $\mathcal{C}$ es $d(\mathcal{C}) = 3$.

**Teorema 3** (Decodificación de máxima verosimilitud)**.** *Es posible corregir hasta*

$$t := \left\lfloor \frac{d(\mathcal{C}) - 1}{2} \right\rfloor$$

*errores, donde* $d(\mathcal{C})$ *denota la distancia mínima del código* $\mathcal{C}$.

**Demostración.** Usando la decodificación de máxima verosimilitud, un vector $y \in \mathbb{F}^n$ es decodificado en una palabra código $c \in \mathcal{C}$, que es cercana a $y$ con respecto a la distancia de Hamming. Formalmente, $y$ es decodificado en una palabra código $c \in \mathcal{C}$ tal que $d(c, y) \leq d(c', y)$, $\forall c' \in \mathcal{C}$. Si hay varios $c \in \mathcal{C}$ con esta propiedad, se elige uno arbitrariamente.

Si la palabra código $c \in \mathcal{C}$ fue enviada y no han ocurrido más de $t$ errores durante la transmisión, el vector recibido es

$$y = c + e \in \mathbb{F}^n,$$

donde $e$ denota al vector error. Esto satisface

$$d(c, y) = d(e, 0) \leq t,$$

y por lo tanto $c$ es el único elemento de $\mathcal{C}$ que se encuentra en una bola de radio $t$ alrededor de $y$. Un decodificador de máxima verosimilitud produce este elemento $c$, y así se obtiene el código correcto. □

**Definición 11.** El *peso Hamming* $\mathrm{wt}(\mathbf{x})$ de un vector $\mathbf{x} \in \mathbb{F}_q^n$ se define como el número de coordenadas no nulas en $\mathbf{x}$.

*Ejemplo* 6. Sea $\mathbf{x} = 2001021 \in \mathbb{F}_3^7$ un vector, entonces su peso Hamming es $wt(\mathbf{x}) = 4$.

El siguiente resultado nos muestra la relación entre la distancia y el peso.

**Teorema 4.** *Si* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, *entonces* $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x} - \mathbf{y})$. *Si* $\mathcal{C}$ *es un código lineal, entonces la distancia mínima d coincide con el peso mínimo de las palabras código no nulas de* $\mathcal{C}$.

**Demostración.** Sean $x, y \in \mathbb{F}_q^n$, por la definición de distancia de Hamming tenemos que $d(x, y) = wt(x - y)$. Se supone ahora que $C$ es un código lineal, luego para todo $x, y \in \mathcal{C}$, $x - y \in \mathcal{C}$, luego para cualquier par de elementos $x, y \in \mathcal{C}$, existe $z \in \mathcal{C}$ tal que $d(x, y) = wt(z) \geq wt(\mathcal{C})$, donde $wt(\mathcal{C})$ es el peso mínimo de $\mathcal{C}$. Por tanto, $d \geq wt(\mathcal{C})$. Por otro lado, para todo $x \in \mathcal{C}$, se tiene que $wt(x) = d(x, 0)$. Como $\mathcal{C}$ es lineal, $0 \in \mathcal{C}$, luego $d(x, 0) \geq d$. Entonces, $wt(\mathcal{C}) \geq d$. Se concluye que $wt(\mathcal{C}) = d$, como se quería. □

Como consecuencia de este teorema, para códigos lineales, la distancia mínima también se denomina *peso mínimo* de un código. Si se conoce el peso mínimo $d$ de un $[n, k]$ código, se denota por un $[n, k, d]$ código.

Se ha demostrado que el problema de calcular la distancia mínima de un código lineal binario es NP-difícil, y el problema de decisión correspondiente es NP-completo. Esto es, formalmente, dada una matriz binaria $H$ de dimensión $m \times n$ y un número entero $w > 0$, saber si existe un vector no nulo $x \in \mathbb{F}_2^n$ de peso menor que $w$ tal que $Hw^T = 0$ es un problema NP-completo.

**Demostración.** Para ello, haremos uso de una transformación polinomial del problema de Decodificación de Máxima Verosimilitud al problema de Distancia Mínima.

El problema de Decodicificación de Máxima Verosimilitud es NP-completo y consiste en dados una matriz binaria $H$ de dimensión $m \times n$, un vector $s \in \mathbb{F}_2^m$ y un número entero $w > 0$. ¿Existe un vector $x \in \mathbb{F}_2^n$ de peso menor o igual que $w$ tal que $Hx^t = s$?

El problema de Decodificación de Máxima Verosimilitud sigue siendo NP-completo bajo ciertas restricciones, luego reformularemos este problema como la versión de campo finito de Suma de Subconjuntos, un problema NP-completo conocido. Además, calcular la distancia

mínima para la clase de códigos lineales sobre un cuerpo de característica 2 es NP-difícil, y el problema de decisión correspondiente Distancia Mínima sobre $GF(2^m)$, abreviado $MD_{2^m}$, es NP-completo. Luego esta prueba se basa en una transformación polinomial de Decodificación de Máxima Verosimilitud a $MD_{2^m}$. Sin embargo, esto no prueba que Distancia Mínima sea NP-completo, ya que el posible conjunto de entradas a Distancia Mínima es un pequeño subconjunto del conjunto de posibles entradas a $MD_{2^m}$. Para ello, se construye una aplicación del código $\mathbb{C}\#$ sobre $GF(2^m)$ a un código binario $\mathbb{C}$, de tal forma que la distancia mínima de $\mathbb{C}\#$ puede determinarse a partir de la distancia mínima de $\mathbb{C}$. Dado que la longitud de $\mathbb{C}$ está acotada por la longitud de un polinomio de $\mathbb{C}\#$, y el mapeo en sí se puede lograr en tiempo polinomial, esto completa la prueba de la NP-completitud de Distancia Mínima. □

Como hemos visto, la distancia mínima es importante en un código lineal. Sin embargo, calcular este parámetro para un código dado puede resultar realmente duro. A continuación presentaremos un algoritmo para el cálculo de la distancia.

---

**Input:** matriz generadora $G_1 = (I_k|A_1)$ de $\mathcal{C}$
**Output:** distancia mínima $\bar{d}_i$

1 $m \longleftarrow 2$
2 $k_1 \longleftarrow k$
3 **while** $rank\ (A_m) \neq 0$ **do**
4 Aplicar la eliminación Gaussiana y posibles permutaciones de las columnas de la matriz $A_{m-1}$ desde $G_{m-1} = \left( \begin{array}{c|c|c} A'_{m-1} & \begin{array}{c} I_{k_{m-1}} \\ \hline 0 \end{array} & \begin{array}{c} A_{m-1} \\ \hline 0 \end{array} \end{array} \right)$ para obtener la matriz generadora $G_m = \left( \begin{array}{c|c|c} A'_m & \begin{array}{c} I_{k_m} \\ \hline 0 \end{array} & \begin{array}{c} A_m \\ \hline 0 \end{array} \end{array} \right)$
5 **end**
6 $C_0 \longleftarrow \{0\}$
7 $i \longleftarrow 0$
8 **while** $\bar{d}_i > d'_i$ **do**
9 $i \longleftarrow i + 1$
10 $C_i \longleftarrow C_{i-1} \cup \bigcup_{j=1}^{m} \{v \cdot G_j : v \in \mathbb{F}(q)^k,\ wt(v) = i\}$
11 $\bar{d}_i \longleftarrow min\{wt(c) : c \in C_i,\ c \neq 0\}$
12 $d'_i \longleftarrow \sum_{j=1, k-k_j \leq i}^{m} (i+1) - (k - k_j)$
13 **end**

---

**Algoritmo 1**: Algoritmo de Brouwer-Zimmermann: cálculo de la distancia mínima de un $[n, k]$ código lineal $\mathcal{C}$.

**Definición 12.** Sea $A_i$, también denotada por $A_i(\mathcal{C})$, el número de palabras código con peso $i$ en $\mathcal{C}$. Se dice que la lista $A_i$ para $0 \leq i \leq n$ es la distribución del peso o espectro del peso de $\mathcal{C}$.

## 2.5 CLASIFICACIÓN POR ISOMETRÍA

Como hemos visto, las propiedades de codificación de un código dependen principalmente de las distancias de Hamming entre diferentes palabras codificadas y entre palabras codificadas y no codificadas. Además, puede ser que un código pueda relacionarse con otro por medio de una aplicación que conserve las distancias de Hamming. De esta forma, podemos definir una relación de equivalencia entre dos códigos que preservan la distancia de Hamming.

Tenemos que dos $(n,k)$-códigos $C, C' \subseteq H(n,q)$ son de la misma cualidad si existe un mapeo

$$\iota : H(n,q) \to H(n,q)$$

con $\iota(C) = C'$ que preserva la distancia de Hamming, es decir,

$$d(w,w') = d(\iota(w), \iota(w')), \qquad \forall w, w' \in H(n,q).$$

Los mapeos con la propiedad anterior se llaman *isometrías*.

**Definición 13.** Dos códigos lineales $C, C' \subseteq H(n,q)$ se llaman *isométricos* si existe una isometría de $H(n,q)$ que aplica $C$ sobre $C'$.

Las permutaciones de las coordenadas son isometrías, que se denominan *isometrías permutacionales*.

**Definición 14.** Sea $S_n$ el grupo isométrico en el conjunto $X = n = \{0, ..., n-1\}$. Dos códigos lineales $C, C' \subseteq H(n,q)$ son isométricos permutacionalmente si existe una isometría permutacional de $H(n,q)$ que aplica $C$ sobre $C'$. Esto es, hay una permutación $\pi$ en el grupo simétrico $S_n$ tal que

$$C' = \pi(C) = \{\pi(c) : c \in C\}, \quad \text{and} \quad d(c,\tilde{c}) = d(\pi(c), \pi(\tilde{c})), \qquad \forall c, \tilde{c} \in C,$$

donde

$$\pi(c) = \pi(c_0, ..., c_{n-1}) := \left(c_{\pi^{-1}(0)}, ..., c_{\pi^{-1}(n-1)}\right)$$

# 3

# CÓDIGOS DE GOPPA

Hay que completar esta introducción

Goppa describió una nueva clase de códigos de corrección de errores lineales no cíclicos. Lo más importante es que algunos de estos códigos excedían el límite asintótico de Gilbert-Varshamov, una hazaña que muchos teóricos códigos pensaban que nunca podría lograrse. En este capítulo vamos a hablar sobre los códigos de Goppa y sus propiedades más importantes.

El desarrollo de este capítulo se ha basado en [2] y [6].

## 3.1 CÓDIGOS REED-SOLOMON

Antes de estudiar los códigos de Goppa, en esta sección introduciremos los códigos Reed-Solomon, que abreviaremos como códigos RS.

**Definición 15.** Para $k \geq 0$, $\mathcal{P}_k$ denota el conjunto de polinomios de grado menor que $k$, incluyendo el polinomio nulo, en $\mathbb{F}_q[x]$. Si $\alpha$ es la n-ésima raíz primitiva de la unidad en $\mathbb{F}_q$ donde $0 \leq k \leq n = q - 1$, entonces el código

$$\mathrm{RS}_k = \left\{ \left( f(1), f(\alpha), ..., f(\alpha^{q-2}) \right) : f \in \mathcal{P}_k \right\}$$

es un $[n, k, n - k + 2]$ código Reed-Solomon sobre $\mathbb{F}_q$.

### 3.1.1 *Decodificación de los códigos Reed-Solomon*

En esta sección presentaremos dos algoritmos para decodificar por el vecino más cercano los códigos Reed-Solomon. En concreto, estudiaremos los métodos de decodificación de Peterson-Gorenstein-Zierler y de Sugiyama. Este primer método consiste en cuatro pasos, en concreto el segundo de ellos destaca por ser muy complicado y llevar demasiado tiempo. Como alternativa a ejecutar este segundo paso, surge el algoritmo de Sugiyama, una aplicación simple del algoritmo de Euclides para polinomios.

#### 3.1.1.1 *Algoritmo de decodificación de Peterson-Gorenstein-Zierler*

Supongamos que recibimos el mensaje $y(x)$, que difiere de la palabra código $c(x)$ en un máximo de $t$ coordenadas. El objetivo de los algoritmos de decodificación es determinar el mensaje original $c(x)$ a partir de $y(x)$. Tenemos que $y(x) = c(x) + e(x)$ donde $c(x) \in \mathcal{C}$ y $e(x)$ denota a los *errores* con peso $\nu \leq t$. Supongamos que el error ocurre en las coordenadas desconocidas $k_1, ..., k_\nu$. Por lo tanto, los errores tendrán la siguiente forma

$$e(x) = e_{k_1} x^{k_1} + \cdots + e_{k_\nu} x^{k_\nu}.$$

Si determinamos $e(x)$, lo que equivale a encontrar el número de errores $\nu$, las posiciones de los errores $k_j$ y los valores de los errores $e_{k_j}$, podemos decodificar el vector recibido como $c(x) = y(x) - e(x)$.

Comenzamos definiendo el *síndrome* $S_i$ de $y(x)$ como el elemento $S_i = y\left(\alpha^i\right)$ para $1 \leq i \leq 2t$. El primer paso del algoritmo será determinar los síndromes. Además, los síndromes satisfacen que

$$S_i = y(\alpha^i) = c(\alpha^i) + e(\alpha^i) = 0 + e(\alpha^i) = \sum_{j=1}^{\nu} e_{k_j} (\alpha^i)^{k_j} = \sum_{j=1}^{\nu} e_{k_j} (\alpha^{k_j})^i, \qquad 1 \leq i \leq 2t.$$

Para $1 \leq j \leq \nu$ simplificaremos la notación, sea $E_j = e_{k_j}$ que denota el *valor del error en la coordenada* $k_j$ y $X_j = \alpha^{k_j}$ denota el *número de la posición del error correspondiente a la posición del error* $k_j$. Así, los síndromes se pueden reformular como sigue.

$$S_i = \sum_{j=1}^{\nu} E_j X_j^i, \qquad 1 \leq i \leq 2t, \tag{1}$$

que resulta en el sistema de ecuaciones:

$$\begin{cases} S_1 = E_1 X_1 + \cdots + E_\nu X_\nu, \\ S_2 = E_1 X_1^2 + \cdots + E_\nu X_\nu^2 \\ \qquad\qquad \vdots \\ S_{2t} = E_1 X_1^{2t} + \cdots + E_\nu X_\nu^{2t} \end{cases} \tag{2}$$

Este sistema es no linear para los $X_j$ y además desconocemos los valores para $E_j$ y $X_j$. Para resolver este sistema, primero obtendremos los $X_j$ con un nuevo sistema lineal que involucrará nuevas variables. Así, el sistema 2 será lineal para los $E_j$ y podremos determinarlos.

Para ello, definimos el *polinomio localizador de errores* como

$$\sigma(x) = (1 - xX_1) \cdots (1 - xX_\nu) = 1 + \sum_{i=1}^{\nu} \sigma_i x^i$$

Observemos que las raíces de $\sigma(x)$ son las inversas de los $X_j$:

$$\sigma\left(X_j^{-1}\right) = 1 + \sigma_1 X_j^{-1} + \cdots + \sigma_\nu X_j^{-\nu} = 0, \qquad 1 \leq j \leq \nu.$$

Multiplicando por $E_j X_j^{i+\nu}$ obtenemos

$$E_j X_j^{i+\nu} + \sigma_1 E_j X_j^{i+\nu-1} + \cdots + \sigma_\nu E_j X_j^i = 0, \qquad \forall i.$$

Si sumamos para todo $j$ en $1 \leq j \leq \nu$ tenemos

$$\sum_{j=1}^{\nu} E_j X_j^{i+\nu} + \sigma_1 \sum_{j=1}^{\nu} E_j X_j^{i+\nu-1} + \cdots + \sigma_\nu \sum_{j=1}^{\nu} E_j X_j^i = 0.$$

En estas sumas hemos obtenido los síndromes de 1, pues $i \geq 1$ y $i + \nu \leq 2t$. Como $\nu \leq t$, tenemos que

$$S_{i+\nu} + \sigma_1 S_{i+\nu-1} + \cdots + \sigma_\nu S_i = 0, \qquad 1 \leq i \leq \nu,$$

que equivale a

$$\sigma_1 S_{i+\nu-1} + \cdots + \sigma_\nu S_i = -S_{i+\nu}, \qquad 1 \leq i \leq \nu.$$

Así, llegamos al siguiente sistemas de ecuaciones del que podemos obtener los $\sigma_k$.

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \cdots & S_\nu & S_{\nu+1} \\ & & \vdots & & \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-2} & S_{2\nu-1} \end{pmatrix} \begin{pmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{2\nu} \end{pmatrix} \tag{3}$$

La dificultad para resolver este sistema está en que desconocemos el valor de $\nu$, que queremos que sea lo más pequeño posible. El siguiente lema nos será útil.

**Lema 1.** *Sea $\mu \leq t$ y sea*

$$\begin{pmatrix} S_1 & \cdots & S_\mu \\ \vdots & \ddots & \vdots \\ S_\mu & \cdots & S_{2\mu-1} \end{pmatrix}$$

*Si $\mu = \nu$, entonces $M_\mu$ no es singular; pero si $\mu > \nu$, entonces $M_\mu$ es singular, donde $\nu$ es el número de errores que se han producido.*

**Demostración.** Ver Lema 5.4.2 en la página 181, [2]. □

Realizaremos un procedimiento iterativo para obtener el valor de $\nu$, empezando con $\mu = t$, que es lo más grande que $\nu$ puede ser. Teniendo en cuenta el lema 1, tenemos que $M_\mu = M_t$.

Luego si $M_\mu$ es singular, reducimos el valor de $\mu$ en 1, $\mu = t - 1$, y volvemos a probar si $M_\mu = M_{t-1}$ es singular. Continuaremos reduciendo $\mu$ en 1 hasta que obtengamos una matriz $M_\mu$ que no sea singular, en cuyo caso obtendremos el valor de $\nu = \mu$. Así, resolvemos el sistema 3 y podremos determinar $\sigma(x)$. Con esto concluimos el segundo paso del algoritmo.

El siguiente paso será determinar las raíces de $\sigma(x)$ y calcular sus inversas para determinar los $X_j$. Esto lo podemos hacer calculando reiteradamente $\sigma(\alpha^i)$ para $0 \leq i < n$. Una vez conocidos los $X_j$, podemos resolver el sistema de ecuaciones 2 para determinar los $E_j$. Ahora ya podremos determinar los valores de $k_j$ y $e_{k_j}$, y así poder obtener el vector error $e(x)$. Finalmente, restando el vector $e(x)$ al vector $y(x)$ podemos determinar el mensaje original $c(x)$, como queríamos.

En resumen, el algoritmo de decodificación de Peterson-Gorenstein-Zierler es el siguiente:

I. Calcular los síndromes $S_i = y(\alpha^i)$ para $1 \leq i \leq 2t$.
II. Iterar decrementando una unidad desde $\mu = t$ hasta que $M_\mu$ no sea singular. Definamos ese valor como $\nu = \mu$ y resolvemos 3 para obtener $\sigma(x)$.
III. Determinar las raíces de $\sigma(x)$ calculando $\sigma(\alpha^i)$ para $0 \leq i < n$. Los $X_j$ coinciden con las inversas de dichas raíces.
IV. Resolver las primeras $\nu$ ecuaciones de 2 para determinar los $E_j$.
V. Calcular los valores de $k_j$ y $e_{k_j}$.
VI. Determinar $c(x)$ como el resultado de la operación $y(x) - e(x)$.

*Ejemplo 7.* Poner un ejemplo chorra

#### 3.1.1.2 *Algoritmo de decodificación de Sugiyama*

El algoritmo de decodificación de Peterson-Gorenstein-Zierler se puede mejorar, en concreto el paso II consume mucho tiempo pues tiene que computar diversos determinantes para cada matriz hasta encontrar alguna que no sea singular. El algoritmo de Sugiyama presenta una alternativa para este segundo paso, de hecho usa el algoritmo de Euclides para determinar el polinomio localizador de errores de una manera más eficiente.

Recordamos que el polinomio localizador de errores $\sigma(x)$ se define como $\prod_{j=1}^{\nu}(1 - xX_j)$. El *polinomio evaluador de errores* $\omega(x)$ se define como

$$\omega(x) = \sum_{j=1}^{\nu} E_j X_j \prod_{\substack{i=1 \\ i \neq j}} (1 - xX_i) = \sum_{j=1}^{\nu} E_j X_j \frac{\sigma(x)}{1 - xX_j}. \tag{4}$$

Observemos que $gr(\sigma(x)) = \nu$ y $gr(\omega(x)) \leq \nu - 1$. Definimos el polinomio $S(x)$ de grado como mucho $2t - 1$ tal que

$$S(x) = \sum_{i=0}^{2t-1} S_{i+1} x^i,$$

donde $S_i$ para $1 \leq i \leq 2t$ son los síndromes.

De esta forma, se cumple la siguiente relación (página 191, [2].)

$$\omega(x) \equiv \sigma(x)S(x) \pmod{x^{2t}},$$

a la que nos referiremos como *ecuacinclave*. Además, tenemos que los polinomios $\sigma(x)$ y $\omega(x)$ son primos relativos.

El algoritmo de Sugiyama es el siguiente.

I. Sean $f(x) = x^{2t}$, $s(x) = S(x)$, $r_{-1}(x) = f(x)$, $r_0(x) = s(x)$, $b_{-1}(x) = 0$ y $b_0(x) = 1$.
II. Iterar incrementando en una unidad desde $i = 1$ hasta $I$, de forma que se cumpla que $gr(r_{I-1}(x)) \geq t$ y $gr(r_I(x)) < t$. En cada iteración habrá que determinar $h_i(x)$, $r_i(x)$ y $b_i(x)$:
$$r_{i-2}(x) = r_{i-1}(x)h_i(x) + r_i(x), \qquad \text{donde } gr(r_i(x)) < gr(r_{i-1}(x)),$$
$$b_i(x) = b_{i-2}(x) - h_i(x)b_{i-1}(x).$$
III. $\sigma(x)$ es un escalar no nulo múltiplo de $b_I(x)$.

Para verificar que este algoritmo funciona, el siguiente lema nos será de utilidad.

**Lema 2.** *Con la notación del algoritmo de Sugiyama, sea $a_{-1}(x) = 1$, $a_0(x) = 0$ y $a_i(x) = a_{i-2}(x) - h_i(x)a_{i-1}(x)$ para $i \geq 1$. Las siguientes afirmaciones son ciertas.*

- $a_i(x)f(x) + b_i(x)s(x) = r_i(x)$ *para* $i \geq -1$.
- $b_i(x)r_{i-1}(x) - b_{i-1}r_i(x) = (-1)^i f(x)$ *para* $i \geq 0$.
- $a_i(x)b_{i-1}(x) - a_{i-1}b_i(x) = (-1)^{i+1}$ *para* $i \geq 0$.
- $gr(b_i(x)) + gr(r_{i-1}(x)) = gr(f(x))$ *para* $i \geq 0$.

**Demostración.** Ver Lema 5.4.11 en la página 191, [2]. □

Comprobemos ahora que el algoritmo de Sugiyama funciona. Por el Lema 2 (i) tenemos que

$$a_I(x)x^{2t} + b_I(x)S(x) = r_I(x). \tag{5}$$

Además, por la ecuación clave sabemos que

$$a(x)x^{2t} + \sigma(x)S(x) = \omega(x) \tag{6}$$

para algún polinomio $a(x)$. Multiplicando 5 por $\sigma(x)$ y 6 por $b_I(x)$ obtenemos

$$a_I(x)\sigma(x)x^{2t} + b_I(x)\sigma(x)S(x) = r_I(x)\sigma(x) \quad \text{y} \tag{7}$$
$$a(x)b_I(x)x^{2t} + \sigma(x)b_I(x)S(x) = \omega(x)b_I(x). \tag{8}$$

Aplicando módulo $x^{2t}$ a ambas ecuaciones tenemos que

$$b_I(x)\sigma(x)S(x) \equiv r_I(x)\sigma(x) \pmod{x^{2t}} \quad \text{y}$$
$$\sigma(x)b_I(x)S(x) \equiv \omega(x)b_I(x) \pmod{x^{2t}}.$$

Por lo tanto,

$$r_I(x)\sigma(x) \equiv \omega(x)b_I(x) \pmod{x^{2t}}. \tag{9}$$

Como $gr(\sigma(x)) \leq t$, por la elección de $I$,

$$gr(r_I(x)\sigma(x)) = gr(r_I(x)) + gr(\sigma(x)) < t + t = 2t.$$

Por el Lema 2 (iv), la elección de $I$ y como $gr(\omega(x)) < t$, entonces se cumple que

$$gr(\omega(x) \times b_I(x)) = gr(\omega(x)) + gr(b_I(x)) < t + gr(b_I(x)) = t + (gr(x^{2t}) - gr(r_{I-1}(x))) \leq 3t - t = 2t.$$

Por 9 tenemos que $r_I(x)\sigma(x) = \omega(x)b_I(x)$. Esto, junto con 7 y 8, implica que

$$r_I(x)\sigma(x) \equiv \omega(x)b_I(x) \pmod{x^{2t}}. \tag{10}$$

Sin embargo, el Lema 2 (iii) afirma que $a_I(x)$ y $b_I(x)$ son primos relativos, y por 10 se cumple que $a(x) = \lambda(x)a_I(x)$. Sustituyendo esta relación en 10,

$$\sigma(x) = \lambda(x)b_I(x). \tag{11}$$

Ahora, sustituyendo estas dos últimas relaciones en 6 obtenemos $\lambda(x)a_I(x)x^{2t} + \lambda(x)b_I(x)S(x) = \omega(x)$. La ecuación 5 implica que

$$\omega(x) = \lambda(x)a_I(x)x^{2t} + \lambda(x)b_I(x)S(x) = \lambda(x) \cdot \big(a_I(x)x^{2t} + b_I(x)S(x)\big) = \lambda(x)r_I(x). \tag{12}$$

Teniendo en cuenta que los polinomios $\sigma(x)$ y $\omega(x)$ son primos relativos y por 11 y 12, $\lambda(x)$ tiene que ser una constante no nula, verificando el paso III del algoritmo de Sugiyama.

Como solo estamos interesados en las raíces de $\sigma(x)$, es suficiente con determinar las raíces de $b_I(x)$ obtenidos en el paso II; lo que nos dará los $X_j$.

Más adelante veremos que este algoritmo sigue funcionando con otras elecciones de $f(x)$ y $s(x)$, con las modificaciones apropiadas de las condiciones sobre las que el algoritmo termina en el paso II. Una de estas modificaciones nos será útil para decodificar los códigos Goppa que veremos a continuación.

## 3.2 CÓDIGOS CLÁSICOS DE GOPPA

Los códigos clásicos de Goppa se introdujeron por V. D. Goppa en 1970. Estos códigos son generalizaciones de códigos BCH y subcódigos de subcuerpos de ciertos códigos GRS.

Para motivar la definición de los códigos de Goppa, introduciremos una construcción de los códigos BCH de longitud $n$ sobre $\mathbb{F}_q$. Sea $t = ord_q(n)$ y sea $\beta$ la raíz enésima primitiva de la unidad en $\mathbb{F}_{q^t}$. Elegimos $\delta > 1$ y sea $\mathcal{C}$ el código BCH de longitud $n$ y distancia $\delta$. Entonces $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_q[x]/(x^n - 1)$ está en $\mathcal{C}$ si y solo si $c(\beta^j) = 0$ para $1 \leq j \leq \delta - 1$. Tenemos que

$$(x^n - 1)\sum_{i=0}^{n-1} \frac{c_i}{x - \beta^{-i}} = \sum_{i=0}^{n-1} c_i \sum_{l=0}^{n-1} x^l \left(\beta^{-i}\right)^{n-1-l} = \sum_{l=0}^{n-1} x^l \sum_{i=0}^{n-1} c_i \left(\beta^{l+1}\right)^i.$$

Como $c(\beta^{l+1}) = 0$ para $0 \leq l \leq \delta - 2$, el lado derecho de la ecuación es un polinomio cuyo término de menor grado tiene grado al menos $\delta - 1$. Por lo tanto, el lado derecho se puede escribir como $x^{\delta-1}p(x)$, donde $p(x)$ es un polinomio en $\mathbb{F}_{q^t}[x]$. Así, podemos decir que $c(x) \in \mathbb{F}_q[x]/(x^n - 1)$ está en $\mathcal{C}$ si y solo si

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \beta^{-i}} = \frac{x^{\delta-1}p(x)}{x^n - 1}$$

o equivalentemente

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \beta^{-i}} \equiv 0 \pmod{x^{\delta-1}}$$

La última equivalencia es la base para la definición de los códigos clásicos de Goppa.

Fijado el cuerpo de extensión $\mathbb{F}_{q^t}$ de $\mathbb{F}_q$, sea $L = \{\gamma_0, ..., \gamma_{n-1}\}$ una tupla de $n$ elementos distintos de $\mathbb{F}_{q^t}$ y sea $g(x) \in \mathbb{F}_{q^t}[x]$ con $g(\gamma_i) \neq 0$ para $0 \leq i \leq n - 1$. Entonces el *código de Goppa* $\Gamma(L, g)$ es el conjunto de vectores $c_0 \cdots c_{n-1} \in \mathbb{F}_q^n$ tal que

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \tag{13}$$

De esta forma, cuando la parte de la izquierda está escrita como una función racional, significa que el numerador es un múltiplo de $g(x)$. Además, trabajar con módulo $g(x)$ es como trabajar en el anillo $\mathbb{F}_{q^t}[x]/(g(x))$, y la hipótesis $g(\gamma_i) \neq 0$ garantiza que $x - \gamma_i$ es invertible en este anillo. Se llama a $g(x)$ el *polinomio de Goppa* de $\Gamma(L, g)$. Notemos que el código BCH de longitud $n$ y la distancia elegida $\delta$ es el código de Goppa $\Gamma(L, g)$ con $L = \{1, \beta^{-1}, ..., \beta^{1-n}\}$ y $g(x) = x^{\delta-1}$.

A continuación, buscaremos una matriz de paridad para $\Gamma(L, g)$. Para ello, observamos que

$$\frac{1}{x-\gamma_i} \equiv -\frac{1}{g(\gamma_i)}\frac{g(x)-g(\gamma_i)}{x-\gamma_i} \pmod{g(x)}$$

ya que, comparando numeradores, $1 \equiv -g(\gamma_i)^{-1}\left(g(x)-g(\gamma_i)\right) \pmod{g(x)}$. Así que por (13) $\mathbf{c} = c_0 \cdots c_{n-1} \in \Gamma(L,g)$ si y solo si

$$\sum_{i=0}^{n-1} c_i \frac{g(x)-g(\gamma_i)}{x-\gamma_i} g(\gamma_i)^{-1} \equiv 0 \pmod{g(x)} \tag{14}$$

Supongamos que $g(x) = \sum_{j=0}^{w} g_j x^j$ con $g_j \in \mathbb{F}_{q^t}$, donde $w = gr(g(x))$. Entonces

$$\frac{g(x)-g(\gamma_i)}{x-\gamma_i} g(\gamma_i)^{-1} = g(\gamma_i)^{-1} \sum_{j=1}^{w} g_j \sum_{k=0}^{j-1} x^k \gamma_i^{j-1-k} = g(\gamma_i)^{-1} \sum_{k=0}^{w-1} x^k \left( \sum_{j=k+1}^{w} g_j \gamma_i^{j-1-k} \right)$$

Por lo tanto, por (14), estableciendo los coeficientes de $x^k$ iguales a 0, en el orden $k = w-1, w-2, ..., 0$, tenemos que $\mathbf{c} \in \Gamma(L,g)$ si y solo si $Hc^T = 0$, donde

$$H = \begin{pmatrix} h_0 g_w & \cdots & h_{n-1} g_w \\ h_0(g_{w-1} + g_w \gamma_0) & \cdots & h_{n-1}(g_{w-1} + g_w \gamma_{n-1}) \\ & \vdots & \\ h_0 \sum_{j=1}^{w} \left( g_j + \gamma_0^{j-1} \right) & \cdots & h_{n-1} \sum_{j=1}^{w} \left( g_j + \gamma_{n-1}^{j-1} \right) \end{pmatrix} \tag{15}$$

con $h_i = g(\gamma_i)^{-1}$.

**Proposición 3.** *La matriz $H$ se puede reducir a una matriz $H'$ de dimensión $w \times n$, donde*

$$H' = \begin{pmatrix} g(\gamma_0)^{-1} & \cdots & g(\gamma_{n-1})^{-1} \\ g(\gamma_0)^{-1}\gamma_0 & \cdots & g(\gamma_{n-1})^{-1}\gamma_{n-1} \\ & \vdots & \\ g(\gamma_0)^{-1}\gamma_0^{w-1} & \cdots & g(\gamma_{n-1})^{-1}\gamma_{n-1}^{w-1} \end{pmatrix} \tag{16}$$

Las entradas de $H'$ están en $\mathbb{F}_{q^t}$. Eligiendo una base de $\mathbb{F}_{q^t}$ sobre $\mathbb{F}_q$, cada elemento de $\mathbb{F}_{q^t}$ se puede representar como un vector columna $t \times 1$ sobre $\mathbb{F}_q$. Reemplazando cada entrada de $H'$ por su correspondiente vector columna, obtenemos una matriz $H''$ de dimensión $tw \times n$ sobre $\mathbb{F}_q$ que tiene la propiedad de que si $\mathbf{c} \in \mathbb{F}_q^n$ está en $\Gamma(L,g)$ si y solo si $H''c^T = 0$.

El siguiente resultado nos muestra los límites en la dimensión y la distancia mínima de un código de Goppa.

**Teorema 5.** *Con la notación de esta sección, sea $\Gamma(L,g)$ un código de Goppa tal que $gr(g(x)) = w$ entonces es un $[n,k,d]$ código con $k \geq n - wt$ y $d \geq w+1$.*

**Demostración.** Las filas de $H''$ pueden ser dependientes, luego esta matriz tiene rango como máximo $wt$. Por lo que $\Gamma(L, g)$ tiene dimensión al menos $n - wt$. Si una palabra código $\mathbf{c} \in \Gamma(L, g)$ tiene peso $w$ o menos, entonces el lado izquierdo de 13 es una función racional, donde el numerador tiene grado $w - 1$ o menos; pero este numerador tiene que ser múltiplo de $g(x)$, lo cual es una contradicción pues el grado de $g$ es $w$. □

**Corolario 1.** *Si $\Gamma(L, g)$ es un código de Goppa tal que $gr(g(x)) = w$, entonces puede corregir hasta*

$$\left\lfloor \frac{w}{2} \right\rfloor$$

*errores.*

**Demostración.** El teorema 3 afirma que es posible corregir hasta

$$\left\lfloor \frac{d(\Gamma(L, g)) - 1}{2} \right\rfloor$$

errores. Además, por el teorema 5 sabemos que la distancia mínima de un código de Goppa $\Gamma(L, g)$ tal que $gr(g(x)) = w$ es $d(\Gamma(L, g)) = w + 1$. Por lo que

$$\left\lfloor \frac{d(\Gamma(L, g)) - 1}{2} \right\rfloor = \left\lfloor \frac{(w + 1) - 1}{2} \right\rfloor = \left\lfloor \frac{w}{2} \right\rfloor.$$

□

### 3.2.1 *Códigos binarios de Goppa*

Los códigos binarios de Goppa son códigos de corrección de errores que pertenecen a la clase de los códigos de Goppa que acabamos de estudiar. La estructura binaria le da más ventajas matemáticas sobre variantes no binarias y, además, tienen propiedades interesantes adecuadas para la construcción del criptosistema McEliece.

**Definición 16.** Fijado el cuerpo de extensión $\mathbb{F}_{2^m}$ de $\mathbb{F}_2$, sea $L = \{\gamma_0, ..., \gamma_{n-1}\} \in \mathbb{F}_{2^m}^n$ una tupla de $n$ elementos distintos de $\mathbb{F}_{2^m}$ y sea $g(x) \in \mathbb{F}_{2^m}[x]$ con $g(\gamma_i) \neq 0$ para $0 \leq i \leq n - 1$. Entonces el *código binario de Goppa* $\Gamma(L, g)$ es el conjunto de vectores $c_0 \cdots c_{n-1} \in \{0, 1\}^n$ tal que

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \tag{17}$$

Observemos que si $g(x)$ es un polinomio irreducible, todos los elementos $\gamma \in \mathbb{F}_{2^m}$ satisfacen $g(\gamma) \neq 0$. A los códigos que cumplan esta propiedad los llamaremos *códigos binarios de Goppa irreducibles*.

En los siguientes resultados se enuncian las propiedades de los códigos binarios de Goppa, que tienen ciertas diferencias con los generales.

**Teorema 6.** *Sea $\Gamma(L, g)$ un código binario de Goppa tal que $gr(g(x)) = w$ entonces es un $[n, k, d]$ código con $k \geq n - wt$ y $d \geq 2w + 1$.*

**Corolario 2.** *Si $\Gamma(L, g)$ es un código de Goppa tal que $gr(g(x)) = w$, entonces puede corregir hasta*

$$\left\lfloor \frac{(2w+1)-1}{2} \right\rfloor$$

*errores.*

Observamos que con estos códigos podemos doblar la capacidad correctora de los códigos generales de Goppa.

### 3.2.2 *Decodificación de los códigos de Goppa*

Como hemos visto, al transmitir una palabra código a un receptor, éste podría recibir la palabra alterada. Para que el receptor pueda determinar el mensaje original, necesitaremos decodificar el mensaje recibido. Supongamos que $E$ es el vector de errores que se añade a la palabra código $C$ transmitida, entonces la palabra recibida $R$ está dada por

$$R = C + E$$

de donde

$$\sum_{\gamma \in L} \frac{R_\gamma}{x - \gamma} = \sum_{\gamma \in L} \frac{C_\gamma}{x - \gamma} + \sum_{\gamma \in L} \frac{E_\gamma}{x - \gamma}.$$

Como $C$ es una palabra código, la primera sumatoria de la parte derecha desaparece al aplicar el módulo $g(x)$, y tenemos que

$$\sum_{\gamma \in L} \frac{R_\gamma}{x - \gamma} = \sum_{\gamma \in L} \frac{E_\gamma}{x - \gamma} \pmod{g(x)}.$$

Definiremos su síndrome como el polinomio $S(x)$ de grado menos que $gr(g(x))$ tal que

$$S(x) = \sum_{\gamma \in L} \frac{R_\gamma}{x - \gamma} \pmod{g(x)}.$$

Acabamos de ver que

$$S(x) = \sum_{\gamma \in L} \frac{E_\gamma}{x - \gamma} \pmod{g(x)}.$$

Sea $M$ un subconjunto de $L$ tal que $E_\gamma \neq 0$ si y solo si $\gamma \in M$. Entonces

$$S(x) = \sum_{\gamma \in M} \frac{E_\gamma}{x - \gamma} \pmod{g(x)}. \tag{18}$$

De esta forma, ahora podemos introducir el polinomio cuyas raíces son las ubicaciones de los errores,

$$\sigma(x) = \prod_{\gamma \in M} (x - \gamma) \tag{19}$$

Sin embargo, para los códigos de Goppa es más conveniente definir una variante de este polinomio de la siguiente forma.

$$\eta(x) = \sum_{\gamma \in M} E_\gamma \prod_{\partial \in M \setminus \{\gamma\}} (x - \partial) \tag{20}$$

Observemos que de esta forma $\sigma(x)$ y $\eta(x)$ deben ser primos relativos.

Derivando la expresión de $\sigma(x)$, tenemos que

$$\sigma'(x) = \sum_{\gamma \in M} \prod_{\partial \in M \setminus \{\gamma\}} (x - \partial) \tag{21}$$

de donde, para cada $\gamma \in M$,

$$\eta(\gamma) = E_\gamma \prod_{\partial \in M \setminus \{\gamma\}} (\gamma - \partial) = E_\gamma \sigma'(\gamma)$$

por lo que $E_\gamma = \frac{\eta(\gamma)}{\sigma'(\gamma)}$. De esta forma, una vez que hemos calculado los polinomios $\sigma$ y $\eta$, las coordenadas del vector error vienen dadas por

$$E_\gamma = \begin{cases} 0 & \text{si} \quad \sigma(\gamma) \neq 0 \\ \\ \frac{\eta(\gamma)}{\sigma'(\gamma)} & \text{si} \quad \sigma(\gamma) = 0 \end{cases}$$

donde $\sigma'(x)$ es la derivada de $\sigma(x)$.

Lo esencial para decodificar los códigos de Goppa es determinar los coeficientes de los polinomios $\sigma$ y $\eta$. Para ello, tenemos que relacionar $\sigma$ y $\eta$ al síndrome de la ecuación 18. Esto se consigue multiplicando las ecuaciones 18 y 19, obteniendo

$$S(x) \cdot \sigma(x) \equiv \eta(x) \pmod{g(x)} \tag{22}$$

La ecuación 22 es la *ecuación clave* para decodificar los códigos de Goppa. Dado $g(x)$ y $S(x)$, el problema de decodificar consiste en encontrar polinomios de grado bajo $\sigma(x)$ y $\eta(x)$ que satisfacen 22.

Reduciendo cada potencia de $x$ (mód $g(x)$) e igualando coeficientes de $1, x, ..., x^{grg-1}$, tenemos que 22 es un sistema de $grG$ ecuaciones lineales donde las incógnitas son los coeficientes de $\sigma$ y $\eta$. Por lo tanto, para probar que el decodificador es capaz de corregir todos los patrones hasta $t$ errores, basta con probar que 22 tiene una única solución con grados de $\sigma$ y de $\eta$ suficientemente pequeños. Esto equivale a que el conjunto de ecuaciones lineales correspondientes sean linealmente independientes.

Supongamos que existen dos pares diferentes de soluciones a 22:

$$S(x)\sigma^{(1)}(x) \equiv \eta^{(1)}(x) \pmod{g(x)} \tag{23}$$

$$S(x)\sigma^{(2)}(x) \equiv \eta^{(2)}(x) \pmod{g(x)} \tag{24}$$

donde $\sigma^{(1)}(x)$ y $\eta^{(1)}(x)$ son primos relativos, al igual que $\sigma^{(2)}(x)$ y $\eta^{(2)}(x)$. Además, $\sigma^{(1)}(x)$ y $g(x)$ no pueden tener ningún factor en común, pues en ese caso ese factor podría dividir a $\eta^{(1)}(x)$, contradiciendo que $\sigma^{(1)}(x)$ y $\eta^{(1)}(x)$ son primos relativos. Así, podemos dividir 23 por $\sigma^{(1)}(x)$ y obtenemos

$$S(x) \equiv \frac{\eta^{(1)}(x)}{\sigma^{(1)}(x)} \pmod{g(x)}$$

de la misma forma para 24,

$$S(x) \equiv \frac{\eta^{(2)}(x)}{\sigma^{(2)}(x)} \pmod{g(x)}$$

de donde,

$$\sigma^{(1)}(x)\eta^{(2)}(x) \equiv \sigma^{(2)}(x)\eta^{(1)}(x) \pmod{g(x)} \tag{25}$$

Si $gr(G) = 2t$ y $gr(\sigma^{(1)}) \leq t$, $gr(\sigma^{(2)}) \leq t$, $gr(\eta^{(2)}) < t$ y $gr(\eta^{(1)}) < t$, entonces se da la siguiente igualdad

$$\sigma^{(1)}(x)\eta^{(2)}(x) = \sigma^{(2)}(x)\eta^{(1)}(x) \tag{26}$$

Así, $\sigma^{(1)}$ divide a $\sigma^{(2)}\eta^{(1)}$, y como $\sigma^{(1)}$ y $\eta^{(1)}$ son primos relativos, $\sigma^{(1)}$ tiene que dividir a $\sigma^{(2)}$. Análogamente, $\sigma^{(2)}$ tiene que dividir a $\sigma^{(1)}$. Como ambos son mónicos, se tiene que $\sigma^{(1)} = \sigma^{(2)}$ y así, $\eta^{(1)} = \eta^{(2)}$. Con esto hemos probado que si el grado de $G$ es $2t$, entonces 22 tiene una única solución cuando $gr(\eta) < gr(\sigma) \leq t$, luego el correspondiente sistema de ecuaciones lineales donde las incógnitas son los coeficientes de $\sigma$ y $\eta$ tiene que ser no singular. En el siguiente teorema se concluye este resultado.

**Teorema 7.** *Si $gr(g(x)) = 2t$, entonces hay un algoritmo de decodificación algebraica de corrección de $t$ errores para el código $q$-ario de Goppa con el polinomio de Goppa $g(x)$.*

Estudiemos ahora este resultado en el caso binario, primero observamos que ya que todos los $E_\gamma$ distintos de cero son iguales a 1, entonces 20 y 21 coincidan. De esta forma, la ecuación 25 ahora pasa a ser

$$\sigma^{(1)}\left(\sigma^{(2)}\right)' \equiv \sigma^{(2)}\left(\sigma^{(1)}\right)' \pmod{g(x)}$$

Ahora, cuando $\sigma$ sea par escribiremos en su lugar $\hat{\sigma}$, mientras que cuando $\sigma$ sea impar escribiremos en su lugar $x\sigma'$. Así, tenemos que

$$\begin{aligned}\left(\hat{\sigma}^{(1)} + x\sigma^{(1)'}\right)\sigma^{(2)'} &\equiv \left(\hat{\sigma}^{(2)} + x\sigma^{(2)'}\right)\sigma^{(1)'} \\ \hat{\sigma}^{(1)}\sigma^{(2)'} + \hat{\sigma}^{(2)}\sigma^{(1)'} &\equiv 0 \pmod{g(x)}.\end{aligned}$$

El lado izquierdo es un cuadrado perfecto, pues todos los polinomios de ese lado son pares. Esto implica que

$$\hat{\sigma}^{(1)}\sigma^{(2)'} + \hat{\sigma}^{(2)}\sigma^{(1)'} \equiv 0 \pmod{\bar{G}(x)}$$

donde $\bar{G}(x)$ es múltiplo de $g(x)$ de menor grado ya que $\bar{G}$ es un cuadrado perfecto. Por lo que, si $gr(\bar{G}) = 2t$, $gr(\sigma^{(1)}) \leq t$ y $gr(\sigma^{(2)}) \leq t$, entonces

$$\hat{\sigma}^{(1)}\left(\sigma^{(2)}\right)' = \hat{\sigma}^{(2)}\sigma^{(1)'}.$$

Por la primalidad relativa, $\sigma^{(1)} = \sigma^{(2)}$. En el siguiente teorema se concluye este resultado.

**Teorema 8.** *Si $gr(g(x)) = t$ y si $g(x)$ no tiene factores irreducibles repetidos, entonces hay un algoritmo de decodificación algebraica de corrección de t errores para el código binario de Goppa con el polinomio de Goppa $g(x)$.*

#### 3.2.2.1 *El algoritmo de decodificación de Sugiyama*

En esta sección presentaremos una modificación del algoritmo de Sugiyama, ya estudiado para los códigos RS, para adaptarlo a los códigos Goppa.

Dado $g(x)$ un polinomio de Goppa con grado $2t$, el algoritmo de decodificación de Sugiyama para los códigos Goppa es el siguiente.

I. Calcular el síndrome $S(x)$.
II. Sean $r_{-1}(x) = g(x)$, $r_0(x) = S(x)$, $U_{-1}(x) = 0$ y $U_0(x) = 1$.
III. Buscar $q_i(x)$ y $r_i(x)$ aplicando el algoritmo de Euclides para encontrar el máximo común divisor de $r_{-1}(x)$ y $r_0(x)$ para $i = 1, ..., k$, hasta que $k$ cumpla que $gr(r_{k-1}(x)) \geq t$ y $gr(r_k(x)) < t$:

$$r_{i-2}(x) = r_{i-1}(x)q_i(x) + r_i(x), \qquad gr(r_i(x)) < gr(r_{i-1})(x)$$

IV. Calcular $U_k(x)$, donde

$$U_i(x) = q_i(x)U_{i-1}(x) + U_{i-2}(x)$$

V. La solución viene dada por:

$$\eta(x) = (-1)^k \delta r_k(x)$$
$$\sigma(x) = \delta U_k(x)$$

**Proposición 4.** *Sea e el número de errores que ocurren realmente y sea k el número de iteraciones del algoritmo descrito, entonces* $k \leq e$.

# CONCLUSIÓN

Añadir conclusión

# BIBLIOGRAFÍA


[1] C. Shannon. *A mathematical theory of communication*. University of Illinois Press, 1948. ISBN 978-0-252-72546-3.

[2] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2010. ISBN 978-0-521-13170-4. URL http://www.cambridge.org/9780521782807.

[3] Alexander Vardy. The intractability of computing the minimun distance of a code. *IEEE Transactions on Information Theory*, 43(6), 1997.

[4] Anton Betten, Michael Braun, Harald Fripertinger, Adalbert Kerber, Axel Kohnert, and Alfred Wassermann. *Error-Correcting Linear Codes*. Springer, 2006. ISBN 978-3-540-28371-5. URL http://www.cambridge.org/9780521782807.

[5] Podestá and Ricardo. Introducción a la teoría de códigos autocorrectores, 2006. URL https://www.famaf.unc.edu.ar/documents/940/CMat35-3.pdf.

[6] Elwyn R. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5), 1973.