



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



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

*Granada, a 28 de febrero de 2022*



---

## ÍNDICE GENERAL

---

Resumen	4
Summary	5
Introducción	6
1. PRELIMINARES	7
1.1. Anillos . . . . .	7
1.2. Cuerpos finitos . . . . .	8
1.3. Polinomios . . . . .	8
1.4. Algoritmos genéticos . . . . .	9
1.5. Clases de complejidad . . . . .	10
2. INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS LINEALES	11
2.1. Introducción . . . . .	11
2.2. Códigos lineales . . . . .	12
2.3. Código dual . . . . .	14
2.4. Pesos y distancias . . . . .	15
2.5. Clasificación por isometría . . . . .	17
2.6. Algoritmo para el cálculo de la distancia . . . . .	18
3. CÓDIGOS DE GOPPA	24
3.1. Códigos clásicos de Goppa . . . . .	24
3.1.1. Códigos binarios de Goppa . . . . .	26
3.1.2. Decodificación de los códigos de Goppa . . . . .	27
Conclusión	34
Bibliografía	35

---

## RESUMEN

---

Añadir resumen

PALABRAS CLAVE:

---

## SUMMARY

---

Añadir resumen en inglés

KEYWORDS:

---

## INTRODUCCIÓN Y OBJETIVOS

---

Añadir introducción

---

## 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  $A \times A \rightarrow A$  denotadas por la suma (denotada por  $+$ ) y producto (denotado por  $\cdot$ ) que verifican los siguientes axiomas:

- Propiedad asociativa de la suma:

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

- Existencia del elemento neutro para la suma:

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

- Existencia del elemento inverso para la suma:

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

- Propiedad conmutativa de la suma:

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

- Propiedad asociativa del producto:

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

- Propiedad distributiva del producto:

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

- Existencia del elemento neutro para el producto:

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

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

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

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 se 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  $f = a_n + a_{n-1}X^{n-1} + \dots + a_1X + a_0 \neq 0$  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

$$gr(f + g) \leq \max(gr(f), gr(g)),$$

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

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

$$gr(f + g) = \max(gr(f), gr(g))$$

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.

---

## 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 general, en una dirección. Este mensaje es una secuencia finita de elementos de un alfabeto dado y es enviado por un *canal de comunicación*, a través del cual es posible que la información se altere por las interferencias y el ruido, lo que se conoce como *ruido del canal*. Es por esto que 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*). Esta manipulación consiste en proteger el mensaje original, ya sea por ejemplo añadiendo redundancia o repitiéndolo, para que posteriormente se pueda corregir el ruido hasta cierto punto. 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. 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.

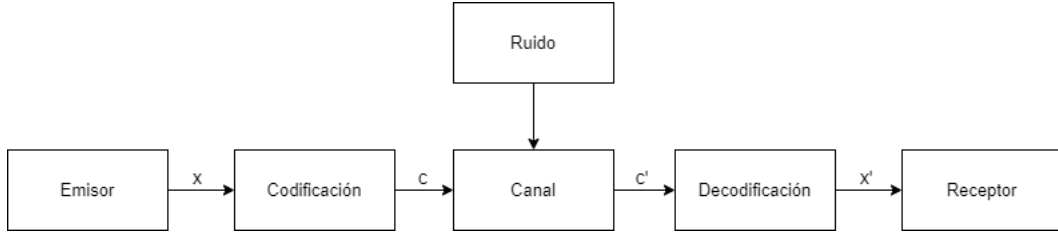


Figura 1: Esquema del modelo de comunicación

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*  $d$  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 transmitir rápida y eficientemente, que detecten y corrijan simultáneamente la mayor cantidad de errores posibles y que haya algoritmos de decodificación eficientes y efectivos. Por lo que habrá que encontrar un balance entre estas distintas metas, pues suelen ser contradictorias entre sí.

## 2.2 CÓDIGOS LINEALES

Sea  $F_q$  el cuerpo finito con  $q$  elementos, denotamos por  $\mathbb{F}_q^n$  el espacio vectorial de las  $n$ -tuplas sobre el cuerpo finito  $\mathbb{F}_q$ . Generalmente los vectores  $(a_1, \dots, a_n)$  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  $\mathcal{C} = \{21, 02, 10, 20\}$ .

Si  $\mathcal{C}$  es 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. Al imponer linealidad sobre los códigos, nos permite conseguir algoritmos de codificación y decodificación más eficientes que otros códigos. 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 pues si realizamos un cambio de base del código podemos obtener otra matriz generadora distinta. 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  $\mathcal{C}$  es una matriz que verifica que

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

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  $H = (-A^T | I_{n-k})$  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  $\mathcal{C}$ .  $\square$

*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

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

Este código se denomina el  $[7,4]$  código de Hamming.

### 2.3 CÓDIGO DUAL

Sabemos que  $\mathcal{C}$  es un subespacio de un espacio vectorial, por lo que podemos calcular el 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

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

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  $\mathcal{C}^\perp$ .

**Demostración.** No sé hacerlo □

De la proposición anterior se deduce lo siguiente.

**Proposición 3.**  $\mathcal{C}^\perp$  es un  $[n, n - k]$  código.

**Demostración.** Sabemos que, por ser  $G$  una matriz de paridad de  $\mathcal{C}^\perp$ ,

$$\mathcal{C}^\perp = \left\{ x \in \mathbb{F}_q^k : Gx^T = 0 \right\},$$

o sea  $\mathcal{C}^\perp$  es el espacio solución de  $k$  ecuaciones con  $n$  incógnitas. Luego, como  $G$  tiene rango  $k$ , hay  $n - k$  variables libres, por lo tanto  $\dim \mathcal{C}^\perp = n - k$ . □

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

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

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

## 2.4 PESOS Y DISTANCIAS

A la hora de corregir errores es importante establecer una medida que nos establezca cuánto de diferentes son las palabras enviadas y recibidas. En este apartado estudiaremos esta idea y cómo puede influir a la teoría de códigos.

**Definición 10.** La *distancia de Hamming*  $d(\mathbf{x}, \mathbf{y})$  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  $d(\mathbf{x}, \mathbf{y})$  satisface las siguientes propiedades.

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

**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  $x \neq z$  no puede ocurrir que  $x = y = z$ , luego  $y \neq x$  o  $y \neq z$ . Es decir, se cumple que  $d(x, y) \neq 0$  o  $d(y, z) \neq 0$ . Entonces por la no negatividad se da la desigualdad. En el caso en el que  $x = z$ , tendríamos que  $d(x, z) = 0$  y también se da la afirmación.  $\square$

Diremos que la *distancia mínima* de un código  $\mathcal{C}$  es la distancia más pequeña de todas las distancias entre dos palabras distintas del 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

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

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.

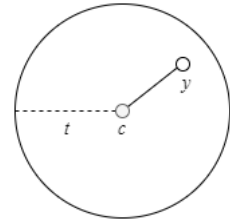
Si la palabra código  $c \in \mathcal{C}$  fue enviada y no han ocurrido más de  $t$  errores durante la transmisión, el vector recibido es

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

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.  $\square$

**Definición 11.** El *peso Hamming*  $wt(\mathbf{x})$  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  $d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} - \mathbf{y})$ . 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  $\mathcal{C}$  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.  $\square$



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 dice entonces que es un  $[n, k, d]_q$  código.

En el artículo [3] 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.

**Teorema 5.** *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  $Hx^T = 0$  es un problema NP-completo.*

Veamos un esquema de la demostración de este resultado. Para ello, haremos uso de una transformación polinomial del problema de Decodificación de Máxima Verosimilitud al problema de Distancia Mínima.

El problema de Decodificación 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  $C\#$  sobre  $GF(2^m)$  a un código binario  $C$ , de tal forma que la distancia mínima de  $C\#$  puede determinarse a partir de la distancia mínima de  $C$ . Dado que la longitud de  $C$  está acotada por la longitud de un polinomio de  $C\#$ , y el mapeo en sí se puede lograr en tiempo polinomial, esto completa la prueba de la NP-completitud de Distancia Mínima.

**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 un código dependen principalmente de las distancias de Hamming entre sus palabras 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.

Sean  $C$  y  $C'$  dos  $[n, k]_q$  códigos, se dice que son de la misma cualidad si existe una aplicación

$$\iota : F_q^n \rightarrow F_q^n$$

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

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

Las aplicaciones 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  $X = n = \{0, \dots, n-1\}$ . 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

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

donde

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

## 2.6 ALGORITMO PARA EL CÁLCULO DE LA DISTANCIA

Como hemos visto, la distancia mínima es importante en un código lineal. Sin embargo, calcular este parámetro para un código dado puede resultar laborioso. A continuación presentaremos el algoritmo de BZ para el cálculo de la distancia, que tiene eficiencia exponencial. Este algoritmo destaca por ser el algoritmo más rápido para calcular la distancia mínima de un código lineal. En particular, si el código es binario se considera eficaz.

---

**Input:** matriz generadora  $G_1 = (I_k | A_1)$  de  $\mathcal{C}$

**Output:** distancia mínima  $\bar{d}_i$

```

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

```

---

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

Veamos, en efecto, que este algoritmo determina la distancia mínima. Consideramos  $\mathcal{C}$  un  $[n, k]_q$  código lineal y  $G$  una matriz generadora. Como el código es lineal, existen matrices  $M \in M_n(q)$  y  $B \in GL_k(q)$  tales que  $B \cdot G \cdot M^T$  es una matriz generadora en forma estándar. De hecho, usando el método de Gauss, podemos obtener una matriz generadora en forma estándar mediante operaciones elementales en las filas y permutaciones de las columnas. En este caso para realizar las operaciones elementales en las filas multiplicaremos desde la izquierda por una matriz  $B_1 \in GL_k(q)$ , y para permutar las columnas multiplicaremos desde la derecha por la traspuesta de una matriz de permutaciones  $M_{\pi_1}$ , de tal forma que

$$G_1 := B_1 \cdot G \cdot M_{\pi_1}^T = (I_{k_1} | A_1),$$

donde  $k_1 = k$ .

Ahora que tenemos la matriz generadora en forma estándar, necesitaremos transformarla de nuevo. Si  $A_1$  no es una matriz nula ni vacía, su rango será  $k_2$  con  $0 < k_2 \leq k_1$ . Aplicando la eliminación de Gauss, podemos obtener  $k_2$  vectores unitarios diferentes en las  $n - k_1$  columnas. Esto es, podemos multiplicar a  $G_1$  desde la izquierda por una matriz  $B_2 \in GL_k(q)$  y desde la derecha por la traspuesta de una matriz de permutaciones  $M_{\pi_2}$  con  $\pi_2(j) = j$  para  $0 \leq j < k_1$ , obteniendo

$$G_2 := B_2 \cdot G_1 \cdot M_{\pi_2}^T = \left( A'_2 \left| \begin{array}{c|c} I_{k_2} & A_2 \\ \hline 0 & 0 \end{array} \right. \right).$$

De esta forma, obtenemos que la matriz  $A'_2$  es una matriz  $k \times k_1$  y la matriz  $A_2$  es una matriz  $k_2 \times (n - k_1 - k_2)$ . Los ceros indican las matrices nulas.

Una vez que tenemos la matriz generadora de esta forma, realizaremos un procedimiento iterativo aplicando la eliminación de Gauss y posibles permutaciones de las columnas de la matriz  $A_2$  hasta que su rango sea 0. Esto es, sea  $i \geq 2$ , la matriz  $A_i$  tiene rango  $k_{i+1}$  con  $0 < k_{i+1} \leq k_i$ . En cada iteración, obtendremos matrices regulares  $B_{i+1} \in GL_k(q)$ , matrices de permutación  $M_{\pi_{i+1}} \in M_n(q)$  con  $\pi_{i+1}(j) = j$  para  $0 \leq j < k_1 + \dots + k_i$  y matrices generadoras

$$G_{i+1} := B_{i+1} \cdot G_i \cdot M_{\pi_{i+1}}^T = \left( A'_{i+1} \left| \begin{array}{c|c} I_{k_{i+1}} & A_{i+1} \\ \hline 0 & 0 \end{array} \right. \right),$$

donde  $A'_{i+1}$  es una matriz  $k \times (k_1 + \dots + k_i)$  y  $A_{i+1}$  es una matriz  $k_{i+1} \times (n - k_1 - \dots - k_{i+1})$ . Repitiendo este procedimiento, finalmente obtendremos una matriz generadora  $G_m$  tal que

$$G_m = B_m \cdot G_{m-1} \cdot M_{\pi_m}^T = \left( A'_m \left| \begin{array}{c|c} I_{k_m} & A_m \\ \hline 0 & 0 \end{array} \right. \right),$$

donde  $A_m$  es una matriz vacía (que significa que no tiene columnas) o es una matriz nula. Entonces,  $k_1 + \dots + k_m \leq n$  y  $A_m$  tiene  $n - k_1 - \dots - k_m$  columnas. En consecuencia, la matriz generadora  $G_m$  tiene  $n - k_1 - \dots - k_m$  columnas nulas y por lo cual todos los elementos del código generado por  $G_m$  tiene peso menor que  $k_1 + \dots + k_m$ .

Definimos para  $1 \leq i \leq k$  los siguientes conjuntos:

$$C_i := \bigcup_{j=1}^m \left\{ v \cdot G_j : v \in \mathbb{F}_q^k, \text{wt}(v) \leq i \right\}$$

Claramente, estos conjuntos forman una cadena ascendente

$$C_1 \subseteq \dots \subseteq C_k,$$

y por lo tanto los pesos mínimos

$$\bar{d}_i := \min \{ \text{wt}(c) : c \in C_i, c \neq 0 \}$$

forman una secuencia decreciente

$$\bar{d}_1 \geq \dots \geq \bar{d}_k = \text{dist}(C).$$

En la mayoría de los casos, no será necesario calcular todos estos valores. En efecto, primero calcularemos  $\bar{d}_1$ . Después, si  $\bar{d}_i$  ya ha sido calculada para algún  $i \geq 1$ , lo compararemos con  $\underline{d}_i$ , que es un límite inferior para el peso de los elementos en  $C \setminus C_i$ . Si  $\bar{d}_i \leq \underline{d}_i$ , hemos terminado. Si no, repetiremos el mismo proceso para  $\bar{d}_{i+1}$ .

Para calcular los límites inferiores para los pesos en los complementos  $C \setminus C_i$ , elegimos un elemento  $c \in C \setminus C_i$ . Como  $c \notin C_i$ , entonces existe, para cada  $j$ , un vector  $v^{(j)} \in \mathbb{F}_q^k$  tal que

$$c = v^{(j)} \cdot G_j, \quad 1 \leq j \leq m, \quad \text{y} \quad \text{wt}(v^{(j)}) \geq i + 1.$$

Para estimar el peso de  $c$ , nos basaremos en los distintos lugares de información en  $G_i$ , es decir, las columnas que contienen la matriz identidad  $I_{k_j}$ . Estas son las columnas de índice  $r$  para  $k_i + \dots + k_{j-1} \leq r < k_1 + \dots + k_j$ . Nos interesan las  $k_j$  coordenadas  $c_r$  de  $c = v^{(j)} \cdot G_j$  correspondientes a estas  $k_j$  columnas. Como  $v^{(j)}$  tiene longitud  $k$ , estas entradas de  $c$  contribuyen al menos el valor de  $i + 1 - (k - k_j)$  al peso de  $c$ . Además, como los conjuntos son disjuntos, para diferentes  $j$ , tenemos que

$$\text{wt}(c) \geq \sum_{j=1}^m (i + 1 - (k - k_j)).$$

Restringiendo esta suma a sumandos positivos, obtenemos el límite inferior

$$\text{wt}(c) \geq \sum_{j: k - k_j \leq i} (i + 1 - (k - k_j)) =: \underline{d}_i.$$

La secuencia de estos límites es incremental, pues el primer sumando es  $i + 1$ :

$$2 \leq \underline{d}_1 < \dots < \underline{d}_k.$$

Además,

$$\underline{d}_k = m + \sum_{j=1}^m k_j.$$

Finalmente, como  $\text{wt}(c) \leq k_1 + \dots + k_m$  para todo  $c \in C$ , existe un índice  $i_0$  tal que

$$\bar{d}_{i_0} \leq \underline{d}_{i_0}.$$

Para este  $i_0$  se cumple que

$$\bar{d}_{i_0} := \min \{ \text{wt}(c) : c \in C_{i_0}, c \neq 0 \},$$

y se sigue cumpliendo que

$$\underline{d}_{i_0} \leq \min \{wt(c) : c \in C \setminus C_{i_0}\}.$$

Por lo que

$$\bar{d}_{i_0} = dist(C),$$

y la palabra código de peso  $dist(C)$  se encuentra en  $C_{i_0}$ .

*Ejemplo 7.* Consideramos el  $[7,3]$  código binario  $\mathcal{C}$  con matriz generadora

$$G_1 = \left( \begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array} \right).$$

Aplicaremos el algoritmo de BZ para calcular la distancia mínima de este código.

Observamos que la matriz generadora  $G_1$  ya está en forma estándar y su conjunto de información es  $\{0,1,2\}$ . Luego lo primero que haremos será aplicar el método de Gauss y permutaciones de las columnas para transformar esta matriz generadora en una matriz generadora de la forma

$$\left( \begin{array}{c|c|c} A'_2 & I_{k_2} & A_2 \\ \hline & 0 & 0 \end{array} \right),$$

para alguna matriz  $A'_2$  de dimensión  $3 \times k_1$  y alguna matriz  $A_2$  de dimensión  $k_2 \times (7 - k_1 - k_2)$ . En nuestro caso, tenemos que

$$G_2 := \left( \begin{array}{ccc|ccc|c} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{array} \right).$$

Esta matriz generadora tiene conjunto de información  $\{3,4,5\}$ . El siguiente paso será realizar de nuevo la eliminación de Gauss y posibles permutaciones de las columnas de la matriz  $A_2$  hasta que su rango sea 0. El algoritmo nos dará la siguiente matriz generadora

$$G_3 := \left( \begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{array} \right).$$

Esta matriz generadora tiene conjunto de información  $\{6\}$ . El conjunto  $C_1$  está formado por las filas de las tres matrices generadoras  $G_1$ ,  $G_2$  y  $G_3$ . Ahora, calculamos el peso de cada uno de ellos, esto es, el número de coordenadas no nulas de cada elemento y obtenemos que todos tienen peso 4, por lo que  $\bar{d}_1 = 4$ . El límite inferior del peso mínimo de los vectores que no

pertenecen a  $C_1$  es  $\underline{d}_1 = 1 + 1 - (3 - 3) + 1 + 1 - (3 - 3) = 4$ . Concluimos que  $d = \bar{d}_1 = 4$  es la mínima distancia de  $C$ .

---

## CÓDIGOS DE GOPPA

---

### Hay que completar esta introducción

En [6], 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 CLÁSICOS DE GOPPA

Fijado el cuerpo de extensión  $\mathbb{F}_{q^t}$  de  $\mathbb{F}_q$ , sea  $L = \{\gamma_0, \dots, \gamma_{n-1}\}$  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

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

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



$$\sum_{i=0}^{n-1} c_i \frac{g(x) - g(\gamma_i)}{x - \gamma_i} g(\gamma_i)^{-1} \equiv 0 \pmod{g(x)} \quad (2)$$

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

$$H = \begin{pmatrix} h_0 g_w & \cdots & h_{n-1} g_w \\ h_0 (g_{w-1} + g_w \gamma_0) & \cdots & h_{n-1} (g_{w-1} + g_w \gamma_{n-1}) \\ \vdots & & \vdots \\ h_0 \sum_{j=1}^w (g_j + \gamma_0^{j-1}) & \cdots & h_{n-1} \sum_{j=1}^w (g_j + \gamma_{n-1}^{j-1}) \end{pmatrix} \quad (3)$$

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

**Proposición 4.** La matriz  $H$  se puede reducir a una matriz  $H'$  de dimensión  $w \times n$ , donde

$$H' = \begin{pmatrix} g(\gamma_0)^{-1} & \cdots & g(\gamma_{n-1})^{-1} \\ g(\gamma_0)^{-1} \gamma_0 & \cdots & g(\gamma_{n-1})^{-1} \gamma_{n-1} \\ \vdots & & \vdots \\ g(\gamma_0)^{-1} \gamma_0^{w-1} & \cdots & g(\gamma_{n-1})^{-1} \gamma_{n-1}^{w-1} \end{pmatrix} \quad (4)$$

Las entradas de  $H'$  están en  $\mathbb{F}_{q^t}$ . Eliendo 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  $H''\mathbf{c}^T = 0$ .

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

**Teorema 6.** 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 1 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  $w$ .  $\square$

**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 6 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

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

□

### 3.1.1 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 15.** Fijada la extensión de cuerpos  $\mathbb{F}_{2^m}$  de  $\mathbb{F}_2$ , sea  $L = \{\gamma_0, \dots, \gamma_{n-1}\} \in \mathbb{F}_{2^m}^n$  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

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \quad (5)$$

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 7.** 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  $d \geq 2w + 1$ .

**Demostración.**

□

**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.

### 3.1.2 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

$$S(x) = \sum_{\gamma \in M} \frac{E_\gamma}{x - \gamma} \pmod{g(x)}. \quad (6)$$

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

$$\sigma(x) = \prod_{\gamma \in M} (x - \gamma) \quad (7)$$

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

$$\eta(x) = \sum_{\gamma \in M} E_{\gamma} \prod_{\partial \in M \setminus \{\gamma\}} (x - \partial) \quad (8)$$

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

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

$$\sigma'(x) = \sum_{\gamma \in M} \prod_{\partial \in M \setminus \{\gamma\}} (x - \partial) \quad (9)$$

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

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

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 6. Esto se consigue multiplicando las ecuaciones 6 y 7, obteniendo

$$S(x) \cdot \sigma(x) \equiv \eta(x) \pmod{g(x)} \quad (10)$$

La ecuación 10 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 10.

Reduciendo cada potencia de  $x$  (mód  $g(x)$ ) e igualando coeficientes de  $1, x, \dots, x^{grg-1}$ , tenemos que 10 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 [10](#) 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 [10](#):

$$S(x)\sigma^{(1)}(x) \equiv \eta^{(1)}(x) \pmod{g(x)} \quad (11)$$

$$S(x)\sigma^{(2)}(x) \equiv \eta^{(2)}(x) \pmod{g(x)} \quad (12)$$

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 [11](#) por  $\sigma^{(1)}(x)$  y obtenemos

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

de la misma forma para [12](#),

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

de donde,

$$\sigma^{(1)}(x)\eta^{(2)}(x) \equiv \sigma^{(2)}(x)\eta^{(1)}(x) \pmod{g(x)} \quad (13)$$

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

$$\sigma^{(1)}(x)\eta^{(2)}(x) = \sigma^{(2)}(x)\eta^{(1)}(x) \quad (14)$$

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 [10](#) 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 8.** 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 8 y 9 coincidan. De esta forma, la ecuación 13 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 9.** 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  $g(x)$ .

### 3.1.2.1 Algoritmo de decodificación de Sugiyama

El algoritmo de Sugiyama es una aplicación simple del 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  $\prod_{j=1}^v (1 - xX_j)$ . El polinomio evaluador de errores  $\omega(x)$  se define como

$$\omega(x) = \sum_{j=1}^v E_j X_j \prod_{\substack{i=1 \\ i \neq j}}^v (1 - xX_i) = \sum_{j=1}^v E_j X_j \frac{\sigma(x)}{1 - xX_j}. \quad (15)$$

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

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

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 *ecuación clave* o *key equation*. 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  $b_i(x)$ :

$$\begin{aligned} r_{i-2}(x) &= r_{i-1}(x)h_i(x) + r_i(x), & \text{donde } gr(r_i(x)) < gr(r_{i-1}(x)), \\ b_i(x) &= b_{i-2}(x) - h_i(x)b_{i-1}(x). \end{aligned}$$

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

$$a_I(x)x^{2t} + b_I(x)S(x) = r_I(x). \tag{16}$$

Además, por la ecuación clave sabemos que

$$a(x)x^{2t} + \sigma(x)S(x) = \omega(x) \tag{17}$$

para algún polinomio  $a(x)$ . Multiplicando 16 por  $\sigma(x)$  y 17 por  $b_I(x)$  obtenemos

$$a_I(x)\sigma(x)x^{2t} + b_I(x)\sigma(x)S(x) = r_I(x)\sigma(x) \quad \text{y} \tag{18}$$

$$a(x)b_I(x)x^{2t} + \sigma(x)b_I(x)S(x) = \omega(x)b_I(x). \tag{19}$$

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

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

Por lo tanto,

$$r_I(x)\sigma(x) \equiv \omega(x)b_I(x) \pmod{x^{2t}}. \quad (20)$$

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 1 (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 20 tenemos que  $r_I(x)\sigma(x) = \omega(x)b_I(x)$ . Esto, junto con 18 y 19, implica que

$$r_I(x)\sigma(x) \equiv \omega(x)b_I(x) \pmod{x^{2t}}. \quad (21)$$

Sin embargo, el Lema 1 (iii) afirma que  $a_I(x)$  y  $b_I(x)$  son primos relativos, y por 21 se cumple que  $a(x) = \lambda(x)a_I(x)$ . Sustituyendo esta relación en 21,

$$\sigma(x) = \lambda(x)b_I(x). \quad (22)$$

Ahora, sustituyendo estas dos últimas relaciones en 17 obtenemos

$$\lambda(x)a_I(x)x^{2t} + \lambda(x)b_I(x)S(x) = \omega(x).$$

La ecuación 16 implica que

$$\omega(x) = \lambda(x)a_I(x)x^{2t} + \lambda(x)b_I(x)S(x) = \lambda(x) \cdot (a_I(x)x^{2t} + b_I(x)S(x)) = \lambda(x)r_I(x). \quad (23)$$

Teniendo en cuenta que los polinomios  $\sigma(x)$  y  $\omega(x)$  son primos relativos y por 22 y 23,  $\lambda(x)$  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.

En esta sección preentaremos una modificación del algoritmo de Sugiyama, ya estudiado para los códigos RS, para adaptarlo a los códigos Goppa. Esta modificación consiste en aplicar el algoritmo de Euclides para encontrar el máximo común divisor de  $r_{i-1}$  y  $r_i$  para  $i = 1, \dots, I$  hasta que  $I$  cumpla que  $gr(r_{I-1}) \geq t$  y  $gr(r_I(x)) < t$ .

---

## 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 minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6), 1997.
- [4] Anton Betten, Michael Braun, Harald Friepertinger, 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.