Grado en Matemáticas

Facultad de Ciencias

VNiVERSiDAD
D SALAMANCA

CRIPTOGRAFÍA BASADA EN CÓDIGOS

Trabajo de Fin de Grado

Julio 2024

Autor:
Juan Sánchez Blázquez

Tutor:
José Ignacio Iglesias Curto

Grado en Matemáticas

Facultad de Ciencias

VNiVERSiDAD D SALAMANCA

CRIPTOGRAFÍA BASADA EN CÓDIGOS

Trabajo de Fin de Grado

Julio 2024

Autor:

Juan Sánchez Blázquez

Tutor:

José Ignacio Iglesias Curto

VNiVERSiDAD D SALAMANCA
CAMPUS DE EXCELENCIA INTERNACIONAL

Facvltad D Ciencias
VNiVERSiDAD D SALAMANCA

Certificado de los tutores TFG Grado en Matemáticas

D. José Ignacio Iglesias Curto, profesor/a del Departamento de Matemáticas de la Universidad de Salamanca,

HACEN CONSTAR:

Que el trabajo titulado "*Criptografía basada en códigos*", que se presenta, ha sido realizado por D. Juan Sánchez Blázquez, con DNI ****2016T y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Matemáticas en esta Universidad.

Salamanca, a fecha de firma electrónica.

Firmado digitalmente por IGLESIAS CURTO JOSE IGNACIO - 70866916Y
Fecha: 2024.07.04 20:23:20 +02'00'

Fdo.: José Ignacio Iglesias Curto

Índice general

Introducción

En palabras del criptógrafo estadounidense Bruce Schneier, *"La criptografía es una ciencia secreta que se preocupa por métodos para enviar mensajes de tal forma que solo el emisor y el receptor puedan entenderlos"* [43]. Esta afirmación sirve para darnos una primera impresión acerca de la importancia de tener sistemas criptográficos seguros. Se conoce la criptografía como el estudio de los métodos que permiten a dos partes comunicarse a través de un canal abierto en un contexto de desconfianza, de modo que el contenido de su comunicación sólo sea accesible para las partes autorizadas [15]. La criptografía se centra en proteger la información, incluso aunque el atacante tenga acceso al mensaje encriptado que queremos transmitir. Busca salvaguardar la integridad de las claves usadas en el proceso de comunicación. Estas claves pueden ser públicas o privadas, es decir, accesibles para todo el mundo o conocidas sólo por las partes que intervienen en el proceso de comunicación respectivamente. Esto lleva a la distinción entre criptografía simétrica, en la que ambas partes conocen una misma clave secreta, y la criptografía asimétrica, en la que ambas partes tienen conocimiento de la clave pública de la otra pero mantienen en secreto sus propias claves privadas. De hecho, tanto la clave pública como la privada están relacionadas pero no se puede obtener la segunda aún conociendo la primera. Veremos a lo largo de este trabajo principalmente cómo implementar criptosistemas de clave pública, y cómo éstos se pueden emplear para establecer claves seguras dentro de la criptografía simétrica.

En 1978, Robert J. McEliece diseñó por primera vez un criptosistema de clave pública basado en códigos [33]. Este criptosistema, de igual nombre, fue denostado al principio debido a los grandes tamaños de clave con los que había que tratar, recibiendo poco interés por parte de los investigadores. Fue también el punto que marcó el nacimiento de la criptografía basada en códigos. Sin embargo, el rápido incremento sufrido en las últimas décadas en la capacidad computacional disponible y la llegada cada vez más próxima de la computación cuántica nos expone frente a un escenario peligroso. Diversos ataques, siendo el primero en manifestarlo el algoritmo de Shor [9], han demostrado la vulnerabilidad de los criptosistemas predominantes actuales, como son RSA o ElGammal en caso de que la computación cuántica sea viable. Estos esquemas se basan en la dificultad de factorizar números primos grandes o en el problema del logaritmo discreto respectivamente. Estas amenazas se pueden ver aún más potenciadas en un futuro en el que los ordenadores cuánticos parece que serán una realidad. Por ello, debemos haber encontrado una alternativa segura frente a tal poder de cálculo para entonces. Fue esta necesidad la que reavivó el interés por estudiar un criptosistema inventado hace casi medio siglo, el esquema de McEliece. Y con el paso de los años y las investigaciones, permanece intacto antes los ataques cuánticos, demostrando así que está más vigente que nunca.

Para conseguir transformar esta preocupación en resultados tangibles, distintas agencias de seguridad alrededor de todo el mundo, como el *National Institute of Standards and Technology (NIST)* de Estados Unidos, el *Instituto Europeo de Estándares de Telecomunicaciones (ETSI)* o los *Comités de Investigación y Evaluación de Criptografía* de Japón entre otros (se puede consultar un listado en [2]), lanzaron concursos de estandarización para la criptografía postcuántica. El más relevante de todos y el que se ha tomado como estado del arte ha sido el proceso impulsado por el NIST. Desde la creación de este proceso hace 12 años, se han evaluado multitud de propuestas, todas englobadas dentro de la criptografía de clave pública y divididas en dos categorías: para el cifrado e intercambio de claves y para la implementación de una firma digital. A medida que han ido pasando rondas, el número de propuestas consideradas aptas ha ido disminuyendo, hasta llegar a la 4ª ronda del proceso, pudiendo consultar los candidatos de esta ronda en [50]. En 2022 se publicaron los criptosistemas elegidos para la estandarización a nivel global, disponibles en [51]. Sólamente en la criptografía postcuántica actual encontramos la criptografía basada en retículos, basada en ecuaciones multivariables, basada en isogenias o basada en funciones hash. Esto nos da una dimensión sobre la gran importancia de las matemáticas en este tema, y demuestra una vez más la aplicación real de estos conocimientos. A lo largo de este trabajo, nos centraremos en otro área de la criptografía, basada en la Teoría de Códigos. Estudiaremos algunos de los candidatos que han pasado por el proceso de estandarización citado previamente.

La importancia actual que ha tomado la criptografía postcuántica y su relevancia futura son dos de las causas que me han llevado a interesarme en este tema y desarrollarlo a lo largo de este trabajo. Siempre he sentido atracción por tratar de comprender cómo la ingente cantidad de datos sensibles, con la que tratamos a lo largo del día sin darnos ni siquiera cuenta, permanecen seguros ante todas las amenazas que les rodean. Como resultado de este trabajo, he adquirido una comprensión significativamente mayor de los mecanismos mediante los cuales los criptosistemas protegen los datos confidenciales. Y al mismo tiempo he descubierto el prometedor área de la criptografía postcuántica con una gran cantidad de nuevos retos por delante. Para lograrlo, he aplicado los conocimientos y habilidades adquiridos a lo largo de estos años, que han sido fundamentales para conseguir el éxito de este esfuerzo.

Para desarrollar este trabajo lo hemos estructurado en capítulos cuyo contenido desglosaremos brevemente a continuación. En el Capítulo 1 se abordan los conceptos matemáticos necesarios para la comprensión del trabajo. Se definen conceptos relativos a la Teoría de Códigos y se habla sobre la complejidad computacional de distintos problemas relacionados con ésta. A lo largo del Capítulo 2 se detalla la estructura y el funcionamiento de los esquemas de McEliece y Niederreiter, así cómo su equivalencia, distintos parámetros e instanciación de los mismos con distintos códigos. Al llegar al Capítulo 3 veremos la aplicación práctica de los criptosistemas presentados en el capítulo anterior. Repasaremos algunas propuestas del proceso de estandarización del NIST así como ciertas propuestas alternativas. Y por último, en el Capítulo 4 analizaremos distintas nociones de seguridad, para posteriormente estudiar algunos de los ataques conocidos contra la criptografía basada en códigos así como ciertas medidas para mitigarlos.

Capítulo 1

Preliminares matemáticos

Cuando hablamos de criptografía basada en códigos (CBC en adelante), debemos empezar considerando ciertos conceptos relacionados con la Teoría de Códigos antes de profundizar en el tema. Existen multitud de recursos bibliográficos acerca de la Teoría de Códigos como pueden ser [54, 31] además de lo estudiado en su momento en este grado ayudándonos de los apuntes [15]. A lo largo de este capítulo, nos abstendremos de demostrar los resultados de proposiciones o teoremas establecidos, ya que sus resultados no contribuyen sustancialmente al desarrollo de este trabajo. En cada una de las demostraciones citaremos bibliografía en la que se pueden consultar, además de poder hacerlo en los recursos generales que acabamos de citar.

1.1. Conceptos básicos

En esta primera sección, vamos a hablar sobre distintos conceptos de Teoría de Códigos ampliamente estudiados y que podemos encontrar en [54].

Definición 1.1. *Un código lineal $\mathcal{C}$ sobre un cuerpo finito $\mathbb{F}$ es un $\mathbb{F}_q$-subespacio vectorial. Un elemento $c \in \mathcal{C}$ se dice que es una palabra del código [23].*

Si tomamos la aplicación $\psi_{\mathcal{C}} : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$, conocida como aplicación de codificación, tenemos que un código $\mathcal{C}$ se puede caracterizar también como la imagen de dicha aplicación, es decir, $\mathcal{C} = Im\psi_{\mathcal{C}} \subset \mathbb{F}_q^n$. Si dicha aplicación es lineal, diremos entonces que el código $\mathcal{C}$ es lineal.

Sea $\mathcal{C}$ un código lineal.

Definición 1.2. *Llamaremos longitud del código $\mathcal{C}$ a la longitud de las palabras codificadas o la dimensión de $\mathbb{F}_q^n$, y se denotará por n. Llamaremos dimensión del código $\mathcal{C}$, $\dim \mathcal{C} = k$, a la dimensión de $\mathcal{C} = Im\psi_{\mathcal{C}}$ como $\mathbb{F}_q$-subespacio vectorial de $\mathbb{F}_q^n$.*

Al tomar la aplicación $\psi_{\mathcal{C}}$ como una aplicación lineal es posible deducir el siguiente concepto.

Definición 1.3. *Se conoce como matriz generadora del código $\mathcal{C}$, $\mathbf{G}$, a la matriz asociada a la aplicación de codificación $\psi_{\mathcal{C}}$.*

Para el propósito de nuestro trabajo, tomaremos como matriz generadora una matriz con k filas y n columnas, pues es lo más usado en Teoría de Códigos. Entonces para codificar un elemento $v \in \mathbb{F}_q^k$, seguiremos el convenio de filas, multiplicando así por la izquierda, es decir, $c = v \cdot G$.

Observación 1.4. *Debemos tener en cuenta que la base del subespacio que define un código no es única. Análogamente, la matriz asociada a una aplicación tampoco lo es, dependiendo de las bases escogidas en los espacios de partida y llegada. Debido a estas consideraciones, la matriz generadora de un código no va a ser única.*

Sea $\mathbf{G}$ una matriz generadora del código $\mathcal{C}$.

$$\mathbf{G} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix} \tag{1.1}$$

Considerando la matriz generadora G, escogemos un menor no nulo de orden k en G. Supongamos que dicho menor son las k primeras columnas. Si fuese necesario, realizando operaciones básicas de filas o multiplicando por la inversa de esta submatriz cuadrada, obtendríamos la siguiente matriz.

$$\tilde{\mathbf{G}} = \begin{pmatrix} 1 & 0 & \cdots & 0 & \tilde{g}_{1k+1} & \cdots & \tilde{g}_{1n} \\ 0 & 1 & \cdots & 0 & \tilde{g}_{2k+1} & \cdots & \tilde{g}_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \tilde{g}_{kk+1} & \cdots & \tilde{g}_{kn} \end{pmatrix} \tag{1.2}$$

Debido a esta idea podemos dar la siguiente definición.

Definición 1.5. *Una matriz generadora de un código en la que k columnas, normalmente las k primeras, forman la matriz identidad se denomina matriz generadora sistemática del código.*

Observación 1.6. *La matriz generadora sistématica de un código $\mathcal{C}$ sí es única.*

Es necesario tener en cuenta que la mayoría de los códigos utilizados a lo largo de este trabajo destacan por su capacidad para corregir errores. Para cuantificar la capacidad de corrección de dichos códigos, se establece una distancia que permita medir en cuánto difieren dos palabras del código [31].

Definición 1.7. *La distancia de Hamming entre dos vectores $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$ es el número de componentes diferentes entre dichos vectores.*

$$d_H(u, v) := |\{i \mid u_i \neq v_i\}| = |\{i \mid u_i - v_i \neq 0\}| \tag{1.3}$$

Definición 1.8. *La distancia mínima de un código es el mínimo de las distancias entre dos palabras diferentes del código.*

Observación 1.9. *En otros trabajos relacionados con Teoría de Códigos podemos encontrar distintas métricas como la métrica de rango o la métrica de Lee [2]. En este trabajo sin embargo utilizaremos la métrica de Hamming por ser la más empleada en el ámbito de la criptografía basada en códigos.*

Definición 1.10. *El peso de Hamming de un vector $v = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$ es el número de componentes de v distintas de 0, es decir,*

$$\omega_H(v) = |\{i \mid v_i \neq 0\}| \tag{1.4}$$

Observación 1.11. *Nótese que $\omega_H(v) = d_H(v, 0)$ y $d_H(u, v) = \omega_H(u - v)$.*

Definición 1.12. *El peso mínimo de un código es el mínimo de los pesos de cualquier palabra del código distinta de 0.*

Cuando se recibe un vector que no pertenece al código, esto significa que han ocurrido errores durante su transmisión. En general, se asume que dicho vector recibido contiene el mínimo número de errores posible. Para determinar lo lejos que está dicha palabra recibida del código empleado nos ayudamos de las siguientes ideas.

Definición 1.13. *La distancia de un vector $v \in \mathbb{F}_q^n$ al código $\mathcal{C}$ es el mínimo de las distancias entre v y cualquier palabra del código $c \in \mathcal{C}$.*

Observación 1.14. *Considerando la anterior definición obtenemos los siguientes conceptos.*

- *Un vector $v \in \mathcal{C} \iff d_H(v, \mathcal{C}) = 0$.*
- *Si $d_H(v, C) = t > 0$, diremos que v tiene un error de peso t, es decir, que existe una palabra del código $c \in \mathcal{C}$ tal $v = c + e$ siendo $e \in \mathbb{F}_q^n$ con $\omega_H(e) = t$.*

Definición 1.15. *Dados dos códigos $\mathcal{C}_1$ y $\mathcal{C}_2$ de la misma longitud n se dice que son equivalentes si $\mathcal{C}_2$ se puede obtener a partir de $\mathcal{C}_1$ intercambiando posiciones de las coordenadas y multiplicando cada una de ellas por un valor no nulo fijo y viceversa. Es decir, son equivalentes si existe un vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n) \in (\mathbb{F}_q^*)^n$ y una permutación $\sigma \in \mathcal{S}_n$ que definen el código $\mathcal{C}_2$ de la siguente manera.*

$$\mathcal{C}_2 = \{(\lambda_1 c_{\sigma(1)}, \lambda_2 c_{\sigma(2)}, \ldots, \lambda_n c_{\sigma(n)}) \mid (c_1, c_2, \ldots, c_n) \in \mathcal{C}_1\}$$

Los códigos lineales se caracterizan por su longitud, dimensión y distancia mínima. Por ello, se dice que un código de longitud n, dimensión k y distancia mínima d es de tipo $[n, k, d]$ (o $[n, k, d]_q$ si queremos especificar que es un código sobre $\mathbb{F}_q$).

Cuando nos referimos a corregir errores contenidos en las palabras de un código, es importante considerar las siguientes nociones.

Definición 1.16. *Un código $\mathcal{C}$ tiene capacidad de detección s si detecta cualquier error de peso s pero no detecta algún error de peso $s + 1$.*

Definición 1.17. *Un código $\mathcal{C}$ tiene capacidad de corrección t si corrige cualquier error de peso t pero no corrige algún error de peso $t + 1$.*

A la hora de medir dichas capacidades para un determinado código, haremos uso del siguiente teorema.

Teorema 1.18 ([54]). *Sea $\mathcal{C}$ un código con distancia mínima d. Entonces $\mathcal{C}$ tiene capacidad de detección $s = d - 1$ y capacidad de corrección $t = \lfloor \frac{d-1}{2} \rfloor$.*

Observación 1.19. *Así pues, la distancia mínima del código determina su capacidad de detección y corrección de errores. Por otro lado, la dimensión mide la cantidad de información en cada vector codificado. Mediante la tasa de información, $R = \frac{k}{n}$, y la distancia relativa, $\delta = \frac{d}{n}$, de un código, podemos dar una medida de las capacidades del mismo y así poder compararlo con otros códigos.*

En ciertas ocasiones se requiere realizar pequeñas modificaciones en los códigos para mejorar alguno de sus parámetros, como pueden ser su capacidad de corrección o su distancia relativa. Para ello podemos servirnos de lo expuesto en la siguiente definición.

Definición 1.20. *Sea $\mathcal{C}$ un código lineal $[n, k, d]$ y $S \subseteq \{1, 2, \ldots, n\}$. El código perforado en las posiciones determinadas por S, denotado $\mathcal{C}^S$, es el conjunto de vectores de longitud $n - |S|$ que se obtiene suprimiendo las coordenadas de las posiciones determinadas por S en cada palabra del código $\mathcal{C}$.*

Por la Definición 1.1 sabemos que un código lineal $\mathcal{C}$ es un subespacio vectorial de $\mathbb{F}_q^n$. Tiene por tanto un subespacio ortogonal $\mathcal{C}^\perp$ respecto del producto escalar habitual.

$$\mathcal{C}^\perp = \{v \in \mathbb{F}_q^n \mid v \cdot c = 0 \ \forall\, c \in \mathcal{C}\}$$

Considerando una base $\{h_1, \ldots, h_{n-k}\}$ de $\mathcal{C}^\perp$, tenemos que $c \in \mathcal{C} \iff c \cdot h_i = 0 \ \forall\, c \in \mathcal{C}$, $\forall\, i = 1, \ldots, n - k$. Por tanto, la matriz $H = (h_1 \mid h_2 \mid \ldots \mid h_{n-k})$ permite determinar de un modo sencillo si un vector v pertenece o no al código, o lo que es lo mismo, si contiene o no errores. Esta matriz H se conoce como matriz de control del código $\mathcal{C}$ y verifica $G \cdot H^t = 0$. Tomaremos en general como matriz de control una matriz de n filas y $n - k$ columnas ya que trabajaremos con matrices de control en orientación vertical.
Como ocurre con la matriz generadora, esta matriz de control no es única dependiendo de la base escogida en el subespacio $\mathcal{C}^\perp$.

Observación 1.21. *Respecto a la matriz de control de $\mathcal{C}$ debemos tener en cuenta la siguiente propiedad.*
Sea $G = \left(Id_{k\times k} | C_{k\times(n-k)}\right)$ la matriz generadora sistemática del código $\mathcal{C}$. Teniendo en cuenta lo que acabamos de hablar, si una matriz $H = \begin{pmatrix} -C_{(k\times(n-k)} \\ Id_{(n-k)\times(n-k)} \end{pmatrix}$ cumple que $G \cdot H = 0$, lo cual teniendo en cuenta la definición de ambas matrices es obvio, tenemos que H es una matriz de control sistemática de $\mathcal{C}$.

Gracias de nuevo a la Definición 1.1 sabemos que por ser $\mathcal{C}^\perp$ un subespacio de $\mathbb{F}_q^n$ puede ser considerado un código lineal. De aquí se deriva el siguiente concepto.

Definición 1.22. *El código dual de $\mathcal{C}$ es el subespacio ortogonal a $\mathcal{C}$. Se denota por $\mathcal{C}^\perp$.*

Observación 1.23. *Con respecto a un código $\mathcal{C}$ y su código dual correspondiente, las siguientes afirmaciones son verdaderas.*

- *Si $\mathcal{C}$ es un código de tipo $[n, k, d]$, entonces $\mathcal{C}^\perp$ es de tipo $[n, n-k, d']$, siendo d' la distancia mínima de $\mathcal{C}^\perp$.*
- *Se verifica que $(\mathcal{C}^\perp)^\perp = \mathcal{C}$. Conocemos que una matriz de control de un código representa una base del subespacio ortogonal escrita como filas mientras que con una matriz generadora ocurre lo mismo para una base del subespacio que conforma el código. Por tanto, si G es una matriz generadora de $\mathcal{C}$ y $\bar{G}$ es una matriz generadora de $\mathcal{C}^\perp$, entonces $\bar{G}^t$ es una matriz de control para $\mathcal{C}$ y G^t lo es para $\mathcal{C}^\perp$.*

Al centrarnos en la corrección de errores nos encontramos con que se emplean distintas técnicas para llevarla a cabo. La corrección de errores consta de varios problemas pero el de decodificar un código lineal aleatorio es uno de los más complicados. Para abordar este tema definiremos el siguiente concepto.

Definición 1.24. *Dado un vector $v \in \mathbb{F}_q^n$, se llama síndrome del vector v a $S(v) = v \cdot H$.*

De esta definición se deduce fácilmente que $S(v) = 0 \iff v \in \mathcal{C}$. La utilidad de conocer el síndrome de un vector reside en que una vez recibimos una palabra cifrada y calculamos su síndrome podemos determinar si efectivamente pertenece al código, en caso de que su síndrome sea nulo, pues esto significa que no contiene ningún error. Si por el contrario, su síndrome tiene un valor $t > 0$, esto significa que la palabra recibida contendrá un error de peso t y podremos corregirlo si la capacidad de corrección del código empleado es mayor o igual que dicho valor.
Debemos saber que es sencillo calcular el síndrome de cualquier palabra de cualquier código. Sin embargo, si no disponemos de más información que ésta, el problema de decodificar un código lineal se convierte en un problema computacionalmente inviable.
Para comprender la razón que nos lleva a tomar en general la palabra del código más cercana a la recibida, es decir, la que menos errores contiene, podemos ayudarnos de la siguiente idea.

Proposición 1.25 ([15])**.** *Sea $\mathcal{C}$ un código lineal con capacidad de corrección t.*

1. *Dado $v \in \mathbb{F}_q^n$ un vector recibido cualquiera con un error e, es decir, $v = c + e$, $c \in \mathcal{C}$, entonces $S(v) = S(e)$.*
2. *Todos los vectores de peso $\leq \lfloor \frac{d-1}{2} \rfloor = t$ tienen síndrome distinto.*

Considerando esto, el síndrome de un vector induce una relación de equivalencia en $\mathbb{F}^n$, de modo que $v \sim v' \iff S(v) = S(v')$. Podemos comprobar que las clases de equivalencia de esta relación son de la forma $v + \mathcal{C} = \{v + c \mid c \in \mathcal{C}\}$. Cada una de ellas se conoce como clase lateral de $\mathcal{C}$ determinada por v. De esa manera, la clase $\bar{0}$ es la clase de equivalencia de las palabras que pertenecen al código. En cada clase sólo puede haber un vector de peso $\leq t$, que es precisamente la capacidad de corrección del código. Y como nos interesa eliminar el error de la palabra recibida, tomaremos como representante de cada clase lateral el vector de peso mínimo. De esa manera, en caso de ser capaces de calcular el síndrome de v podríamos corregir la palabra errónea recibida [47], siempre que como ya hemos dicho, el código tenga una capacidad de corrección suficiente para el error contenido.

Existen distintos algoritmos de decodificación por síndromes que veremos en capítulos posteriores, pero todos se basan en la idea del algoritmo que vamos a presentar a continuación.

Consideramos un código $\mathcal{C}$ con capacidad de corrección t.

Precómputo	Se calcula una tabla con entradas $\{(e, S(e)) \mid e \in \mathbb{F}^n,\ w_H(e) \leq t\}$.
Entrada	Vector recibido $v \in \mathbb{F}^n$
	1. Calcular el síndrome del vector v 2. **si** $S(v) = 0$: $v \in \mathcal{C}$ y se devuelve v como el vector decodificado **si no**: Continuar 3. Se busca $S(v)$ en la tabla de síndromes **si** $S(v)$ está en la tabla: Se recupera el vector error e correspondiente y se devuelve $c = v - e$ **si no**: Se devuelve "Error detectado, no corregible"
Salida	$v \in \mathcal{C}$ o "Error detectado, no corregible"

Tabla 1.1: Algoritmo genérico de decodificación por síndromes

En caso de que el cuerpo considerado no fuese binario, bastaría con calcular la tabla de síndromes para todos los posibles errores de peso $\leq t$ salvo constantes. Así si tuviéramos $S(v) = (s_1, s_2, \ldots, s_{n-k})$ siendo s_j la primera coordenada no nula, bastaría buscar en la tabla el síndrome $s_j^{-1} \cdot S(v)$ y si el error asociado en la tabla es e entonces el error contenido en v sería $s_j \cdot e$.
En algunos casos concretos existen ciertos algoritmos que se basan en la idea de permitir corregir errores más allá de la capacidad de corrección del código que estamos usando y se conocen como algoritmos de decodificación completa. Veremos cómo se pueden aprovechar en algunos criptosistemas basados en códigos más adelante. En cuanto al algoritmo de decodificación por síndromes se sabe que es aplicable a cualquier código, el problema es que el tamaño de la tabla aumenta a medida que aumentan n o d. Por lo tanto, la dificultad de decodificar un código lineal se reduce a resolver el problema de decodificación por síndromes. Este problema se ha demostrado NP-completo como veremos en secciones posteriores y ahí radica su dificultad.

1.2. Familias de códigos

Cuando hablamos de códigos lineales, existen distintas familias de códigos cada una con sus propiedades. Algunas de ellas son las de los códigos BCH, códigos cíclicos, códigos de Goppa, códigos Reed-Solomon, códigos alternantes o códigos Reed-Muller. No profundizaremos sobre todas ellas sólo sobre aquellas con un mayor impacto en el área de la CBC que veremos a continuación. Hay una gran cantidad de bibliografía que ha sido ampliamente estudiada y, si desea profundizar en el tema, puede consultar por ejemplo [13, 27, 31]. Veremos a continuación algunos de los códigos estudiados como posibles candidatos en el escenario futuro de la criptografía postcuántica.

1.2.1. Códigos cíclicos

Esta familia de códigos no se utiliza explícitamente a lo largo del trabajo, pero me parecía importante comentarla pues puede facilitar la comprensión de ciertos conceptos sobre los que iremos hablando más adelante y sobre los que queríamos dar una pequeña base. Para ahondar más en ellos se puede consultar [8].

Definición 1.26. *Un código lineal $\mathcal{C}$ es un código cíclico si permutando cíclicamente las coordenadas de cualquier palabra del código se obtiene otra palabra del código, es decir, si verifica que para toda palabra $c = (c_1, c_2, \ldots, c_{n-1}, c_n) \in \mathcal{C}$ entonces $(c_n, c_1, \ldots, c_{n-1}) \in \mathcal{C}$.*

Esta definición puede interpretarse como una aplicación biyectiva entre el espacio vectorial $\mathbb{F}_q^n$ y el anillo cociente $\mathbb{F}_q[x]/(x^n - 1)$ tomado como espacio vectorial de dimensión n.

$$\begin{array}{ccc} \mathbb{F}_q^n & \xrightarrow{\sim} & \mathbb{F}_q[x]/(x^n-1) \\ (c_0, c_1, \ldots, c_{n-1}) & \longleftrightarrow & c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} \end{array} \tag{1.5}$$

A partir de esta idea obtenemos la siguiente proposición, que nos ayuda a entender la manera en la que se trabaja con las palabras de un código cíclico.

Proposición 1.27 ([27])**.** *Multiplicar por x en $\mathbb{F}_q[x]/(x^n - 1)$ equivale a aplicar una permutación cíclica en $\mathbb{F}_q^n$.*

1.2.2. Códigos de Goppa

En 1970, el matemático ruso *Valery Denisovich Goppa* fue capaz de aprovechar la relación existente entre la Geometría Algebraica y los códigos de manera que éstos se pudieran usar en un proceso de comunicación seguro y eficiente. De esa manera surgieron los códigos de Goppa [47]. Para hablar de estos códigos nos ayudaremos de lo estudiado en [24] y [13].

Definición 1.28. *Sea $g(z) = g_0 + g_1 z + g_2 z^2 + \ldots + g_t z^t \in \mathbb{F}_{q^m}[z]$ y sea $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$, conocido como conjunto de soporte, tal que $g(\alpha_i) \neq 0$, $\forall\, \alpha_i \in L$. Entonces el código definido por*

$$\left\{ c = (c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n : \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} \equiv 0 \ mod\, g(z) \right\} \tag{1.6}$$

es un código de Goppa que se constituye gracias a $g(z)$ y L. Se denota por $\mathcal{C}_\Gamma(L, g(z))$.

Se puede consultar en [47, Teorema 1.17] que $(z - \alpha_i)$ es invertible $mod\, g(z)$. Derivado de este teorema, obtenemos también el siguiente resultado que nos permite identificar al código de Goppa de otro modo.

Corolario 1.29. *Un vector $c \in \mathcal{C}_\Gamma(L, g(z)) \iff \sum_i c_i \left(\frac{g(\alpha_i) - g(z)}{z - \alpha_i}\right) g(\alpha_i)^{-1} = 0$ como polinomio en $\mathbb{F}_{q^m}[z]$.*

Como resultado de dicho teorema obtenemos también el siguiente corolario.

Corolario 1.30. *La matriz de control H de un código de Goppa $\mathcal{C}_\Gamma(L, g(z))$ sobre $\mathbb{F}_{q^m}$ es de la siguiente forma.*

$$\begin{pmatrix} g_t g(\alpha_1)^{-1} & g_t g(\alpha_2)^{-1} & \cdots & g_t g(\alpha_n)^{-1} \\ (g_t\alpha_1 + g_{t-1})g(\alpha_1)^{-1} & (g_t\alpha_2 + g_{t-1})g(\alpha_2)^{-1} & \cdots & (g_t\alpha_n + g_{t-1})g(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ (g_t\alpha_1^{t-1} + \ldots + g_1)g(\alpha_1)^{-1} & (g_t\alpha_2^{t-1} + \ldots + g_1)g(\alpha_2)^{-1} & \cdots & (g_t\alpha_n^{t-1} + \ldots + g_1)g(\alpha_n)^{-1} \end{pmatrix} \tag{1.7}$$

Esta matriz se puede separar en las siguientes matrices.

$$H = \underbrace{\begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix}}_{C} \cdot \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}}_{X} \cdot \underbrace{\begin{pmatrix} g(\alpha_1)^{-1} & 0 & \cdots & 0 \\ 0 & g(\alpha_2)^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g(\alpha_n)^{-1} \end{pmatrix}}_{Y}$$

Un código de Goppa $\mathcal{C}_\Gamma(L, g(z))$ de longitud n es un código lineal sobre $\mathbb{F}_q$. Por la propia definición de $\mathbb{F}_{q^m}$ podemos tomar sus elementos como vectores de longitud m sobre $\mathbb{F}_q$. Así la matriz de control del código de Goppa será una matriz $mt \times n$ con al menos t columnas linealmente independientes sobre $\mathbb{F}_q$. Por ello, la distancia mínima del código resulta ser $d \geq t + 1$. Además, la matriz XY sobre $\mathbb{F}_q$ tiene un máximo de mt columnas linealmente independientes, con lo que su rango será $\leq mt$. Esto se debe a que la matriz X se construye de manera que sus columnas son linealmente independientes sobre $\mathbb{F}_q$ al estar formadas por las potencias de α_i del polinomio de Goppa $g(z)$, que se eligen adecuadamente para garantizar la independencia lineal de las t columnas. Además, al multiplicar por la matriz Y, que es una matriz diagonal con escalares distintos de cero, no se ve afectada la independencia lineal de las mismas. Por tanto, su dimensión será $k \geq n - mt$.

1.2.3. Códigos Low Density Parity Check, Moderate Density Parity Check y sus variantes cuasicíclicas

Como veremos en capítulos posteriores de este trabajo, los códigos de Goppa no son los únicos que están siendo estudiados como posibles candidatos para ser usados en la criptografía postcuántica. Los códigos LDPC y los MDPC son propuestas con un prometedor porvenir debido a que carecen de una estructura algebraica concreta y a un conjunto de propiedades que veremos a continuación [35].

Definición 1.31. *Un código LDPC de parámetros (n, r, w) es un código lineal de longitud n y codimensión r, siendo $r = n - k$, que admite una matriz de control con un peso de fila constante w. Por su lado, un código MDPC $[n, r, w]$ es un código lineal con características idénticas a los LDPC, con la salvedad de que tienen pesos de fila mayores, normalmente de la magintud de $\mathcal{O}(\sqrt{n \log n})$.*

Las filas de las matrices de control de los códigos LDPC de peso constante w cumplen además que w es lo suficientemente bajo como para que la mayoría de sus componentes

sean nulas por lo que estas matrices se denominan matrices dispersas. Existen algoritmos de decodificación que aprovechan esta propiedad de una manera muy eficiente, lo cual es muy provechoso para nuestros intereses.

Existen variantes de los códigos LDPC y MDPC cuyas ventajas veremos más adelante. Para realizar ciertas transformaciones en dichos códigos, sobre todo la de reducir el tamaño de las claves como veremos más adelante, se pueden emplear conceptos como los siguientes.

Definición 1.32. *Una matriz circulante es aquella matriz cuadrada que está definida completamente por su primera fila, al ser cada fila resultado de rotar a la derecha cada uno de los componente de la primera [8].*

Definición 1.33. *Una matriz formada por submatrices o bloques circulantes se conoce como matriz de bloques circulantes.*

Definición 1.34. *Un código lineal $[n, r]$ es cuasicíclico si existe $n_0 \in \mathbb{Z}^+$ tal que al realizar cada desplazamiento cíclico de una palabra del código por n_0 posiciones obtengamos una palabra del código [35].*

Para conseguir las variantes cuasicíclicas de los códigos LDPC y MDPC podemos construir sus matrices generadoras y de control del siguiente modo. Recordemos que n es la longitud del código, o bien un código LDPC o bien uno MDPC. Si existe cierto entero p tal que $n = n_0 p$, podremos tomar como matrices generadora y de control de dicho código matrices formadas por bloques circulantes de tamaño $p \times p$, siendo necesariamente k un múltiplo de p. Aprovecharemos el hecho de que el álgebra de las matrices circulantes $p \times p$, en este caso particular serán binarias, es isomorfa al álgebra de los polinomios $\mod x^p - 1$ sobre $\mathbb{F}_2$. Se debe a que las matrices circulantes están completamente determinadas por sus primeras filas, y a que podemos interpretar una palabra del código como un polinomio debido a la Proposición 1.27. Esto nos permitiría llevar a cabo cálculos de manera eficiente, lo cual es ventajoso en el contexto de la CBC. De esta manera conseguiríamos las variantes cuasicíclicas de los códigos LDPC y MDPC.

1.3. Criptografía de clave pública

En el caso de la criptografía de clave privada, o simétrica, la seguridad estaba garantizada si se conseguía mantener en secreto la clave privada común establecida por ambas partes [17]. Pero existe otro tipo de criptografía, conocida como criptografía asimétrica, en la que las dos partes se comunican empleando para ello pares de claves públicas y privadas. Veamos esquemáticamente cómo consiguen llevar a cabo dicha comunicación [55].

- La parte receptora, llámese *Alice*, establece una clave secreta S_k y una clave pública P_k relacionada de cierta manera con la secreta.
- Publica la clave P_k y mantiene en secreto la clave privada S_k.
- *Bob*, que es la parte emisora del mensaje m, usando la clave publicada por *Alice* encripta el mensaje y obtiene el texto cifrado c.
- Una vez que recibe c, cómo *Alice* conoce la clave secreta necesaria para descifrar el mensaje c y estar ésta relacionada con la clave pública, puede recuperar m.

Para que este esquema sea seguro, un atacante, llamado generalmente *Eve*, no debe ser capaz de recuperar la clave secreta aunque tenga conocimiento de la clave pública empleada para cifrar el mensaje m. Puede tener acceso al mensaje cifrado c pero esto no le debe revelar ninguna información adicional que pueda aprovechar para romper el sistema. El esquema presentado es el adoptado por todos los criptosistemas de clave pública y veremos casos concretos de cómo aplicarlo a lo largo de este trabajo.

Vamos a definir a continuación dos conceptos que aparecen repetidamente en textos criptográficos y nos parece necesario resaltarlos.

Definición 1.35 ([55]). *Una función $f : X \longrightarrow Y$ es una función de puerta trampa si cumple con las siguientes propiedades.*

1. *Es fácil de calcular. Dado un elemento $x \in X$, es computacionalmente fácil de calcular $f(x)$.*
2. *Es difícil de invertir. Dada una salida $y \in Y$, es computacionalmente difícil encontrar un elemento $x \in X$ tal que $f(x) = y$.*
3. *Dispone de una puerta trampa. Existen piezas de información secreta que hacen fácil de calcular la inversa de f. De esa manera, se calcula eficientemente $f^{-1}(y)$ para cualquier $y \in Y$ si se dispone de dicha información.*

Definición 1.36 ([31]). *Una función $f : X \longrightarrow Y$ se dice que es una función hash si cumple las siguientes propiedades.*

1. *Es determinista, es decir, $f(x) = f(x')$ para $x, x' \in X$ si $x = x'$.*
2. *Es difícil de invertir, pues dado un valor $y \in Y$ es difícil encontrar $x \in X$ tal que $f(x) = y$.*
3. *Es resistente a colisiones, es decir, dado $y \in Y$ es difícil encontrar $x, x' \in X$ tales que $f(x) = y = f(x')$.*
4. *Dado $x \in X$, es díficil encontrar $x' \in X$ con $f(x') = f(x)$.*
5. *La salida es de longitud fija, independientemente de la longitud de la entrada.*
6. *Es fácil de calcular $f(x)$ $\forall$ $x \in X$.*

En ambos casos, fácil y difícil se refieren en todo momento a la complejidad computacional de los cálculos.

1.4. Complejidad computacional

Con la aparición de la computación cuántica en el horizonte y el rápido incremento de la capacidad computacional, nos enfrentamos a problemas serios en lo que a seguridad criptográfica se refiere. Estos avances obligarán a implementar nuevos criptosistemas para contrarrestar las amenazas que representan [11].

A lo largo de este trabajo, vamos a tratar de exponer por qué los criptosistemas basados en códigos son una buena alternativa en cuanto a seguridad postcuántica se refiere. Hay dos ideas principales que debemos tener en cuenta a la hora de afianzar ese pensamiento y son los siguientes problemas difíciles de resolver relacionados con la Teoría de Códigos. Por un lado, la dificultad de resolver el problema de decodificación para códigos lineales arbitrarios y por otro lado, la dificultad de recuperar la estructura de un cierto

código a partir de una matriz generadora arbitraria, es decir, obtener un algoritmo de decodificación eficiente [56].
Para facilitar al lector la comprensión de esta subsección, trataremos a continuación algunos conceptos relacionados con la complejidad computacional [7].

Definición 1.37. *La clase P se define como el conjunto de problemas computacionales que pueden resolverse mediante un algoritmo el cual está garantizado que termine su ejecución en un cierto número de iteraciones acotado por un polinomio en la longitud de la entrada.*

Nos referimos a ella como la clase de algoritmos de tiempo polinómico. Los problemas incluidos en esta clase se consideran computacionalmente viables e incluye por ejemplo la resolución de ecuaciones lineales, el problema de ordenar una lista o la multiplicación de matrices.

Definición 1.38. *La clase NP se define como el conjunto de problemas computacionales que pueden ser resueltos empleando algoritmos no deterministas cuyo tiempo de ejecución está acotado por un polinomio en la longitud de la entrada. Se corresponde por tanto a la clase de algoritmos no deterministas en tiempo polinómico.*

Cuando hablamos de algoritmos no deterministas lo estamos haciendo sobre algoritmos que al tener que elegir entre dos alternativas crean dos copias de sí mismos para desarrollar ambos casos. Este hecho puede llevar a un crecimiento exponencial del número de copias. El problema se tomará como resuelto si una de estas copias es capaz de producir la respuesta correcta. Esta clase incluye problemas conocidos como el problema del viajante o el problema del ciclo hamiltoniano.

Tras una extensa investigación, se ha hecho evidente que demsotrar que NP = P es difícil de alcanzar, siendo este uno de los problemas del milenio que continúa sin ser resuelto a día de hoy. Se ha demostrado sin embargo, que si cualquier problema NP posee un algoritmo en tiempo polinómico, entonces el resto de problemas NP también lo tendrían y por tanto se cumpliría la igualdad. Los problemas que cumplen la característica que acabamos de enunciar se conocen como NP-completos (existen multitud de ejemplos en [28]).

1.4.1. Problemas NP-completos en Teoría de Códigos

Teniendo en cuenta lo dicho, probaremos a continuación que el primero de los problemas relacionado con Teoría de Códigos referido anteriormente es NP-completo. Si nos centramos en el segundo de ellos, la indistinguibilidad de un código depende de la clase de códigos que empleemos en los distintos criptosistemas. El mejor candidato son los códigos de Goppa, aunque se han probado otras alternativas con códigos Generalizados de Reed-Solomon (GRS) o códigos de Gabidulin, mostrándose para ambos ciertas vulnerabilidades [56]. La dificultad a la hora de resolver ambos problemas les otorga a los criptosistemas basados en códigos la robustez necesaria frente a los ataques de computadores cuánticos.

Queremos demostrar que el problema de decodificación para códigos lineales arbitrarios es NP-completo, pues esto permite considerar a la criptografía basada en códigos

como un sistema criptográfico seguro, al estar basada en un problema computacionalmente difícil de resolver. Procederemos a presentar formalmente el tema que nos ocupa, proporcionándonos una comprensión completa del contenido expuesto anteriormente en [7].

Problema 1.39 (Problema de decodificación general de un código)**.** *Dados como entrada una matriz binaria A, un vector binario $y \in \mathbb{F}_2^{n-k}$ y un entero no negativo t, se trata de ver si se puede encontrar un vector $x \in \mathbb{F}_2^n$ con peso de Hamming $\leq t$ tal que $x \cdot A = y$.*

Se trata de un problema decisional, es decir, un problema que basado en el conjunto de datos de entrada decide si la propiedad enunciada se cumple o no. Es relativamente sencillo ver que este problema está englobado en los considerados NP. Lo que nosotros queremos demostrar ahora es que efectivamente es NP-completo. Para ello, queremos reducir el problema que acabamos de presentar al problema que enunciaremos a continuación [7].

Problema 1.40 (Problema de emparejamiento tridimensional o problema 3DM)**.** *Tomando como entrada un subconjunto $U \subseteq T \times T \times T$, siendo T un conjunto finito, se quiere encontrar un conjunto $W \subseteq U$ tal que $|W| = |T|$ y tal que no haya dos elementos de W que conicidan en niguna de sus coordenadas.*

Vamos a ver a continuación un ejemplo para reinterpretar así este problema en términos de operaciones de matrices, lo que a posteriori nos va a permitir vincularlo con el Problema 1.39.

Ejemplo 1.41. *Consideraremos el conjunto $T = \{1, 2, 3, 4\}$ y vamos a ver dos ejemplos distintos, tomando en ambos $|U| = 9$. Vamos a representar las tripletas del conjunto U en una matriz de incidencia, que tendrá $|U|$ filas y $3|T|$ columnas, donde cada columna dentro de cada bloque de $|T|$ columnas representará un elemento del conjunto T. Por ello, cada fila de la matriz representará una tripleta del subconjunto U. Por este motivo, vemos fácilmente que cada fila tendrá siempre peso 3, pues dentro de cada bloque solo una columna puede ser no nula. Intentaremos resolver el problema para los diferentes subconjuntos U escogidos.*

1. $U_1 = \{(1,2,1), (1,3,2), (1,1,2), (2,1,4), (2,3,1), (2,2,3), (3,2,1), (3,1,1), (4,4,4)\}$

	1	2	3	4	1	2	3	4	1	2	3	4
$(1,2,1)$	1	0	0	0	0	1	0	0	1	0	0	0
$(1,3,2)$	1	0	0	0	0	0	1	0	0	1	0	0
$(1,1,2)$	1	0	0	0	1	0	0	0	0	1	0	0
$(2,1,4)$	0	1	0	0	1	0	0	0	0	0	0	1
$(3,1,1)$	0	0	1	0	1	0	0	0	1	0	0	0
$(2,3,1)$	0	1	0	0	0	0	1	0	1	0	0	0
$(2,2,3)$	0	1	0	0	0	1	0	0	0	0	1	0
$(3,2,1)$	0	0	1	0	0	1	0	0	1	0	0	0
$(4,4,4)$	0	0	0	1	0	0	0	1	0	0	0	1
Suma mód 2	1	1	1	1	1	1	1	1	1	1	1	1

En este caso, podemos ver que si tomamos las tripletas coloreadas, obtenemos un subconjunto W que cumple el enunciado del Problema 3DM. Además, si sumásemos mód 2 *sus respectivas filas de la matriz de incidencia obtendríamos el vector* $(11\ldots1)$. *Por tanto, la existencia de una solución para el Problema 1.40 equivale a encontrar las* $|T|$ *filas cuya suma* mód 2 *resulte en dicho vector*[1].

2. $U_2 = \{(1,2,1),(1,3,2),(2,1,4),(2,2,3),(3,2,1),(4,4,4),(3,3,3),(2,2,2),(1,1,1)\}$

	1	2	3	4	1	2	3	4	1	2	3	4
$(1,2,1)$	1	0	0	0	0	1	0	0	1	0	0	0
$(1,3,2)$	1	0	0	0	0	0	1	0	0	1	0	0
$(2,1,4)$	0	1	0	0	1	0	0	0	0	0	0	1
$(2,2,3)$	0	1	0	0	0	1	0	0	0	0	1	0
$(3,2,1)$	0	0	1	0	0	1	0	0	1	0	0	0
$(4,4,4)$	0	0	0	1	0	0	0	1	0	0	0	1
$(3,2,3)$	0	0	1	0	0	1	0	0	0	0	1	0
$(2,2,4)$	0	1	0	0	0	1	0	0	0	0	0	1
$(1,4,1)$	1	0	0	0	0	0	0	1	1	0	0	0
Suma mód 2	–	–	–	–	–	–	–	–	–	–	–	–

En este ejemplo no podemos encontrar el subconjunto W deseado de acuerdo con el enunciado del problema que estamos estudiando. O lo que es lo mismo, no existe ninguna combinación de tripletas del conjunto U que al sumar mód 2 *sus respectivas filas de la matriz de incidencia nos dé como resultado el vector* $(11\ldots1)$. *Por tanto, el Problema 3DM no tiene solución en el caso de tomar U_2.*

Teniendo en mente el ejemplo que acabamos de presentar, vamos a ver cómo se lleva a cabo la reducción que nos permite demostrar que el Problema 1.39 es NP-completo [7]. Supongamos que tenemos un algoritmo en tiempo polinómico para el Problema 1.39. Entonces dada una entrada $U \subseteq T \times T \times T$ para el Problema 3DM, tomemos A como la matriz de incidencia descrita en el ejemplo anterior. Cómo se ha visto en el ejemplo, resolver el Problema 3DM equivale a resolver el Problema 1.39, que toma como entradas A, $y = (11\ldots1)$ y $w = |T|$, y aplicando el algoritmo cuya existencia estamos suponiendo descubriríamos en tiempo polinomial si existe el emparejamiento. Con esto, podríamos afirmar que la existencia de un algoritmo polinómico para el problema de decodificación general de un código implica la existencia de un algoritmo en tiempo polinomial para el problema 3DM. De hecho, esto nos llevaría a concluir que existe un algoritmo polinómico para cada problema NP, por lo que podemos asegurar que el Problema 1.39 es NP-completo. Gracias a esta afirmación, sabemos que los criptosistemas basados en códigos son resistentes ante los ataques de los computadores actuales, pues no existen algoritmos que resuelvan el Problema 3DM en tiempo polinómico, y a día de hoy también están haciendo frente a los ataques realizados mediante la computación cuántica.

Anteriormente ya hemos comentado que existe una gran cantidad de problemas NP-completos. Algunos de ellos están relacionados con la Teoría de Códigos (aparecen varios

[1]También se puede pensar en la solución del Problema 3DM como la existencia de un subconjunto W cuyas filas de la matriz de incidencia formen una matriz de permutación dentro de cada bloque en la de incidencia. Recordemos que cada bloque servía para representar una de las coordenadas de la tripleta.

ejemplos en [3], sin embargo no me fue posible acceder a ellos pues la fuente a consultar sólo estaba disponible en ruso). A pesar de que no vamos a profundizar más sobre ellos en este trabajo, me parece interesante tenerlos en cuenta.

- El primero de ellos lo descubrí al investigar el artículo [18]. En él, se presenta el Problema del subcódigo equivalente que resulta ser también NP-completo y está relacionado con el concepto de código equivalente de la Definición 1.15. Basándose en dicho problema los autores muestran la mejora conseguida en un esquema de identificación existente, en este caso el de Girault. Para ello, se ayudan del problema citado y recurren también al Problema 1.39, que ha sido probado NP-completo en esta misma sección de nuestro trabajo.
- Consultando el trabajo realizado en [56], encontré otro de los problemas NP-completos relacionado con los códigos. En él, se demuestra de manera similar a lo hecho anteriormente para el Problema 1.39, que los problemas de encontrar un código perforado con ciertas características y un código perforado equivalente son NP-completos. Podemos encontrar el concepto de código perforado en la Definición 1.20. La dificultad de estos problemas permite corregir algunas vulnerabilidades del esquema de Niederreiter, cuando éste emplea códigos GRS, sobre las que hablaremos más adelante.

Capítulo 2

Criptosistemas basados en códigos

Cuando hablamos sobre criptosistemas basados en códigos, existen esencialmente dos clases, sobre cuya estructura se basan todos los demás criptosistemas que emplean los códigos en su construcción. Por un lado encontramos el criptosistema de McEliece, creado por Robert J. McEliece, sobre el que hemos hablado previamente, y por otro, tenemos el criptosistema de Niederreiter. Vamos a ver a continuación las principales características de ambos, así como sus similitudes y diferencias, teniendo en cuenta que se puede establecer una equivalencia entre ellos. Para explicar las tres primeras secciones de este capítulo, hemos hecho uso de lo visto en [47].

2.1. Esquema de McEliece

En su forma original, este criptosistema fue diseñado para utilizar los códigos de Goppa [33]. Uno de los principales motivos para usar estos códigos y no otros fue el hecho de que poseen un algoritmo de decodificación eficiente. Sin embargo, al aumentar el número de investigaciones, se ha llegado a implementar usando otros códigos, por ejemplo MDPC o su variante cuasicíclica, sobre lo que hablaremos más adelante.

De aquí en adelante, tomaremos como parámetros de los códigos de Goppa: n la longitud del código, k la dimensión del código sobre $\mathbb{F}_q$ y t el grado del polinomio de Goppa. Sus valores son conocidos por lo que formarán parte de la clave pública del criptosistema. Tomaremos a no ser que se indique lo contrario $q = 2$, por lo que trabajaremos en un cuerpo binario. Por otro lado, el soporte L, que es un conjunto de elementos de $\mathbb{F}_{q^m}$, y el poliniomio g que definen al código de Goppa permanecen secretos. Así mismo, se mantendrán ocultas la matriz de permutación P y la matriz invertible de dispersión S, utilizadas para realizar la encriptación del mensaje como veremos a continuación. Tienen también el objetivo de ocultar la estructura de la matriz generadora subyacente G, pues si un atacante fuera capaz de averiguar G no le llevaría demasiado trabajo encontrar un decodificador eficiente para dicha matriz. Por tanto es muy importante conseguir proteger la matriz privada G de la mejor manera posible [38].

2.1.1. Implementación del esquema

Para explicar su funcionamiento consideraremos a *Alice* la parte receptora del mensaje mientras que *Bob* sería en este caso el emisor[1].

1. Generación de claves: *Alice* genera las claves pública (P_k) y privada (S_k) de acuerdo con los valores públicos dispuestos para los parámetros.
   a) *Alice* escoge un código de Goppa binario $[n, k]$ y su matriz generadora, $k \times n$, G. Este código es capaz de corregir hasta t errores.
   b) A continuación selecciona una matriz S binaria aleatoria no singular $k \times k$ y una matriz P de permutación $n \times n$.
   c) Calcula ahora la matriz $k \times n$, $\hat{G} = S \cdot G \cdot P$.
   d) Toma como $P_k = (\hat{G}, t)$ y como $S_k = (S, G, P)$.
2. Encriptación: Supongamos que *Bob* quiere mandar un mensaje cifrado a *Alice*.
   a) *Bob* tiene un mensaje en texto plano m de longitud k.
   b) Toma la clave pública de *Alice*.
   c) Genera un vector aleatorio e de n bits y peso t.
   d) Calcula el texto cifrado $c = m \cdot \hat{G} + e$ y se lo envía a *Alice*.
3. Desencriptación: *Alice* recibe el texto cifrado c y lo desencripta de la siguiente manera.
   a) Calcula P^{-1} usando su clave privada.
   b) Multiplicando por P^{-1}, obtiene $c \cdot P^{-1} = m \cdot S \cdot G + \underbrace{e \cdot P^{-1}}_{peso\, t}$
   c) Como el código de Goppa utilizado permite corregir hasta t errores, *Alice* es capaz de recuperar m. Para ello calcula el síndrome correspondiente al vector $c \cdot P^{-1}$, de modo que detectaría el error $e \cdot P^{-1}$, y lo eliminaría de la palabra cifrada recibida, obteniendo así la palabra cifrada corregida $\bar{c}$. A partir de esto, resuelve el sistema de ecuaciones definido por

$$m \cdot S \cdot G = \bar{c} \equiv (x_1, x_2, \ldots, x_k) \cdot G = (\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n).$$

   Y como $m \cdot S = (x_1, x_2, \ldots, x_k)$, al ser S invertible por cómo la hemos escogido, obtenemos finalmente el mensaje $m = (x_1, x_2, \ldots, x_k) \cdot S^{-1} = (m_1, m_2, \ldots, m_k)$.

Gracias a esta explicación, podemos comprender de forma genérica cómo sucede todo el proceso de intercambio de mensajes al emplear un esquema de McEliece. Este criptosistema se basa tanto en la dificultad de decodificar un código lineal aleatorio, visto gracias al Problema 1.39, como en la dificultad del Problema de indistinguibilidad entre la clave pública, en este caso una matriz generadora, y una matriz aleatoria [44]. Ambos hechos lo hacen realmente interesante como verdadera alternativa para la seguridad postcuántica. Además le otorgan seguridad ante un atacante con poder computacional limitado, por lo que conserva su integridad aún a día de hoy, con distintos niveles de seguridad dependiendo de sus parámetros. Existe cierta reticencia a la hora de adoptarlo como criptosistema predominante debido al gran tamaño de su clave. Sin embargo, su principal ventaja es la

[1] La práctica de referirse a las dos partes como *Alice* y *Bob* es una convención comúnmente empleada en la gran mayoría de los textos analizados que discuten sobre protocolos criptográficos.

de poseer algoritmos eficientes de encriptación y decodificación, por ello en los últimos años han aumentado las investigaciones sobre él [47]. Además, conociendo sólo su clave pública el atacante no puede distinguir el código oculto de un código lineal aleatorio. Se tiene también que una instancia del esquema de encriptación de McEliece puede reducirse a una instancia del Problema 1.39, que como ya sabemos es NP-completo [23].

2.1.2. Proceso de decodificación

Desarrollaremos a continuación cómo se lleva a cabo la decodificación en el esquema de McEliece cuando éste se construye empleando un código de Goppa. Explicaremos para ello ciertos conceptos necesarios.

Sea $y = (y_1, y_2, \ldots, y_n)$ el vector recibido que contiene r errores, asumiendo que $2r+1 \leq d$. Sea $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, tenemos

$$y = (y_1, y_2, \ldots, y_n) = \underbrace{(c_1, c_2, \ldots, c_n)}_{\text{pal. del cód.}} + \underbrace{(e_1, e_2, \ldots, e_n)}_{\text{vector error } e}$$

siendo e un vector de peso r. Para decodificar correctamente el vector recibido, necesitamos realizar las siguientes acciones.

- Localizar las posiciones del error, $B = \{i : 1 \leq i \leq n,\ e_i \neq 0\}$.
- Encontrar los valores de error correspondientes a dichas posiciones.

Y para lograrlo, definimos los siguientes polinomios.

Definición 2.1. *Tenemos el polinomio localizador de errores $l(z)$ y el polinomio evaluador de errores $w(z)$.*

- $l(z) \coloneqq \prod_{i \in B}(z - \alpha_i)$
- $w(z) \coloneqq \sum_{i \in B} e_i \prod_{j \in B,\, j \neq i}(z - \alpha_j)$

Definición 2.2. *Dado y el vector recibido se define su síndrome, $S(y)$, como*

$$S(y) \coloneqq \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i} \mod g(z)$$

Si tomamos ahora y una palabra cualquiera, y pensamos en ella como la palabra compuesta por una palabra del código $c \in \mathcal{C}$ y un error añadido $e \in \mathbb{F}_q^n$, al cumplirse para toda palabra del código que $S(c) = 0$ por la Definición 1.28, entonces resultará que el síndrome de la palabra recibida coincide con el síndrome del vector error añadido.

$$S(y) \coloneqq \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i} = \cancel{\sum_{i=1}^{n} \frac{c_i}{z - \alpha_i}}^{\text{Def.1.28}} + \sum_{B} \frac{e_i}{z - \alpha_i} = \sum_{B} \frac{e_i}{z - \alpha_i} \mod g(z) = S(e)$$

Proposición 2.3. *Sea e el vector error con peso $r \leq \lfloor \frac{t}{2} \rfloor$. Sean $l(z)$, $w(z)$ y $S(y)$ como acabamos de definir. Se cumplen entonces las siguientes propiedades*

1. $gr(l(z)) = r$
2. $gr(w(z)) \leq r - 1$
3. $mcd(l(z), w(z)) = 1$
4. $e_i = \frac{w(\alpha_i)}{l'(\alpha_i)}$ *con* $i \in B$
5. $l(z)S(y) = w(z) \mod g(z)$

Demostración. Tengamos en cuenta las Definiciones 2.1 y 2.2.

1. Por cómo hemos definido $l(z)$, al ser B el conjunto de posiciones de error y saber que el peso del error es r se sigue entonces que $gr(l(z)) = r$.
2. Por el mismo motivo, tenemos que $gr(w(z)) \leq r - 1$.
3. Supongamos que $mcd(l(z), w(z)) = p(z)$ siendo $p(z)$ no trivial $p(z)$. Por tanto, este factor $p(z)$ divide a ambos polinomios. Sabemos que α_i, $i \in B$, son las raíces de $l(z)$, o lo que es lo mismo $l(\alpha_i) = 0 \; \forall i \in B$. Evaluando estas raíces en $w(z)$, de su definición se deduce que $w(\alpha_i) \neq 0$ siempre que $e_i \neq 0$. Tenemos que $p(z)|l(z)$, lo que se cumple si y sólo si $p(\alpha_i) = 0$ para ciertas raíces α_i de $l(z)$, y para ninguna otra. Por otro lado, recordemos que $p(z)|w(z)$. Si para una cierta raíz α_i tal que $p(\alpha_i) = 0$ resulta entonces que $(x - \alpha_i)|p(z)$. Como sabemos que $(x - \alpha_i)|p(z)$ y por su parte $p(z)|w(z)$, esto implica que $(x - \alpha_i)|w(z)$. Teniendo esto, debe cumplirse que $w(\alpha_i) = 0$ para cierto α_i. Por lo tanto llegamos a contradicción, por lo que se deduce que el polinomio $p(z)$ debería ser una constante. Así queda demostrado que $mcd(l(z), w(z)) = 1$.
4. Por definición las posiciones del error se obtienen directamente de las raíces de $l(z)$, teniendo así $B = \{i \,|\, \alpha_i \text{ es una raíz de } l(z)\}$. Por otro lado como $l(z) \coloneqq \prod_{i \in B}(z - \alpha_i)$ su derivada será $l'(z) = \sum_{i \in B} \prod_{j,\, j \neq i}(z - \alpha_j)$. Si tomamos ahora

$$\frac{w(z)}{l'(z)} = \frac{\sum_{i \in B} e_i \prod_{j \in B,\, j \neq i}(z - \alpha_j)}{\sum_{i \in B} \prod_{j \in B,\, j \neq i}(z - \alpha_j)}$$

   De este modo tenemos que $\frac{w(\alpha_i)}{l'(\alpha_i)} = e_i \frac{\prod_{j \in B,\, j \neq i}(z - \alpha_j)}{\prod_{j \in B,\, j \neq i}(z - \alpha_j)} = e_i$
5. Por las definiciones de ambos polinomios se tiene, $\mod g(z)$,

$$l(z)S(y) = \prod_{i \in B}(z - \alpha_i) \sum_{i \in B} \frac{e_i}{z - \alpha_i} = \frac{e_1 \prod_{i \in B}(z - \alpha_i)}{z - \alpha_1} + \ldots + \frac{e_r \prod_{i \in B}(z - \alpha_r)}{z - \alpha_r} =$$
$$= e_1 \prod_{i \in B,\, i \neq 1} (z - \alpha_i) + \ldots + e_r \prod_{i \in B,\, i \neq r} (z - \alpha_i) = \sum_{i \in B} e_i \prod_{j \in B,\, j \neq i} (z - \alpha_j) = w(z)$$

□

Teniendo en cuenta las definiciones dadas y la proposición anterior obtenemos el siguiente algoritmo de corrección de errores para un código de Goppa.

1. Calcular el síndrome $S(y) \coloneqq \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}$.
2. Resolver la ecuación $l(z)S(y) \equiv w(z) \, mod \, g(z)$ teniendo en cuenta que los polinomios $l(z)$ y $w(z)$ son de la forma $l(z) = l_0 + l_1 + \ldots + l_{r-1}z^{r-1} + z^r$ y $w(z) = w_0 + w_1 z + \ldots + w_{r-1}z^{r-1}$ respectivamente. Si el código fuese binario, podríamos usar que $w(z) = l'(z)$.

3. Determinar el conjunto de posiciones de los errores $B = \{i : 1 \leq i \leq n,\, l(\alpha_i) = 0\}$.
4. Calcular el valor de los errores gracias a $e_i = \frac{w(\alpha_i)}{l'(\alpha_i)}\ \forall\, i \in B$. En caso de tomar un código binario, se tendría $e_i = 1\ \forall\, i \in B$.
5. De este modo, el vector error está definido por los valores e_i para $i \in B$ y ceros en el resto de coordenadas.
6. Por último, podemos calcular $c = y - e$.

Tomando este esquema como base se puede construir el algoritmo de Patterson [40], que es el algoritmo conocido que decodifica de una manera más eficiente los códigos de Goppa binarios. Para ello, calcula el síndrome $S(y)$ del vector recibido y resuelve la ecuación $l(z)S(y) \equiv w(z)\, mod\, g(z)$, aprovechando el hecho de ser un código binario. Por este motivo, $w(z) = l'(z)$, lo que permite corregir t errores [23]. Al trabajar sobre un cuerpo de característica 2, el polinomio localizador de errores puede agrupar por un lado las potencias pares de z y por otro las impares obteniendo así $l(z) = a^2(z) + zb^2(z)$.

Entrada	Vector recibido y, código de Goppa $\mathcal{C}_\Gamma(L, g)$
	1. Calcular el síndrome $S(y)$ de un elemento de $\mathbb{F}_{q^m}[z]/\langle g(z)\rangle$
	2. Calcular $T(z) = S(y)^{-1}\, mod\, g(z)$
	3. Calcular $P(z) = \sqrt{T(z) + z}\, mod\, g(z)$
	4. Calcular $a(z)$ y $b(z)$ tales que $a(z) = b(z)P(z)\, mod\, g(z)$
	5. Calcular el polinomio localizador de errores $l(z) = a^2(z) + zb^2(z)$
	6. Hallar las raíces de $l(z)$
	7. Encontrar las posiciones de error, o lo que es lo mismo el vector e
Salida	Vector error e

Tabla 2.1: Algoritmo de Patterson

Vemos claramente como el polinomio $l(z)$ calculado en el paso 5 del algoritmo es el localizador de errores pues $l(z)S(y) = w(z) \mod g(z)$ y entonces $(a^2(z) + zb^2(z))S(y) = b^2(z)$, al ser $l'(z) = w(z)$. Con esto, $\frac{a(z)}{b(z)} = \sqrt{S(y)^{-1} - z} \mod g(z)$.
Una vez se han corregido los posibles errores contenidos en la palabra del código, para encontrar el mensaje enviado basta con resolver

$$(m_1, m_2, \ldots, m_k) \cdot G = (c_1, c_2, \ldots, c_n) \text{ que equivale a } G^t \cdot \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

o lo que es lo mismo, resolver un sistema de n ecuaciones con k incógnitas.

Como hemos mencionado anteriormente, la utilización del algoritmo de Patterson es extensa en el ámbito del descifrado de códigos de Goppa binarios irreducibles. Existen sin embargo otras variantes las que es recomendable al menos que las conozcamos. Una de ellas es la conocida como *list decoding*. Aunque basada en el algoritmo de Patterson, es un tipo de decodificación no determinista que permite corregir errores más allá de la

capacidad de corrección del código[2]. Debemos tener en cuenta que en ocasiones, aunque la capacidad de corrección del código sea t, esto no quiere decir que no pueda corregir algún error de peso $t+1$. Esto sucede en caso de que este error pueda asociarse unívocamente a la única palabra del código más cercana a él [15]. Esto permitiría añadir más de t errores a la hora de encriptar un mensaje lo que dificultaría a su vez el trabajo a los posibles atacantes. La desventaja del *list decoding* es su menor eficiencia, pues necesita recorrer muchas más posibilidades hasta dar con la palabra correcta. Esto es lo que le lleva a no ser considerado como algoritmo alternativo a la hora de decodificar códigos de Goppa.

2.1.3. Asignación de claves

En la versión original[3], los parámetros considerados eran $n = 1024 = 2^{10}$, $t = 50$, $m = 10$ y $k = n - mt = 524$. Aunque con estos parámetros el esquema continúa resistiendo los ataques a día de hoy, algo que no ha sucedido al tratar de usar otros códigos en su construcción, se ha demostrado que puede llegar a ser insuficiente mediante distintos ataques. Por este motivo, en el proceso de estandarización del NIST, se ha recomendado tomar los siguientes valores de los parámetros $n = 6960$, $k = 5413$, $t = 119$ y $m = 13$ para conseguir un nivel de seguridad postcuántica suficiente. Además, sabemos que sus etapas de encriptación y decodificación se llevan a cabo eficientemente, por lo que se postula como una buena alternativa entre los sistemas de seguridad postcuánticos [38]. Sin embargo, los tamaños de P_k y S_k son demasiado grandes, sobre todo porque siguen creciendo a medida que el valor de los parámetros y el nivel de seguridad requerido aumenta. Este es uno de los principales puntos que se esgrime en contra del McEliece cuando se habla sobre la posibilidad de implantarlo como estándar de la seguridad postcuántica. Una manera sencilla de disminuir el tamaño de las claves sería emplear como matriz generadora G del código de Goppa su versión sistemática, con la que sólo habría que almacenar $n - k$ columnas, teniendo así tamaños de P_k de $k(n-k)$ bits y S_k de $(k^2 + (n-k)^2) + (t \times m) + ((n-k) \times m)$ bits, relativos a los tamaños de las respectivas matrices. Resultan así claves públicas y privadas de $1047,74\,KB$ y $3973{,}23\,KB$ respectivamente. Esto supondría por tanto un ahorro significativo de memoria respecto de los tamaños de clave sin usar matrices sistemáticas, en el que los tamaños de las claves son de $k \times n = 4709,31\,KB$ para la pública y de $(k^2 + n^2) + (t \times m) + (n \times m) = 9729{,}47\,KB$ en el caso de la secreta. Ésta es sólo una de las posibilidades consideradas a la hora de reducir el coste, existen otras como la propuesta de Niederreiter que detallaremos a continuación o el uso de otro tipo de códigos sobre lo que hablaremos más adelante.
Si observamos la siguiente tabla se ve claramente cómo al utilizar matrices sistemáticas se reduce sustancialmente el tamaño de la clave pública del esquema, necesitando por ello una menor capacidad de memoria.

[2]En concreto para el caso de los códigos de Goppa binarios irreducibles permite corregir de forma eficiente aproximadamente $n - \sqrt{n(n-2t-2)}$ errores [10].

[3]Tamaños de claves: $P_k = kn \approx 67\,KB$, $S_k = (k^2 + n^2) + (t \times m) + (n \times m) \approx 167\,KB$.

Parámetros	**Tamaño P_k (bytes)**		**Factor de trabajo**	
(n,k,t)	Original	Sistemática	Encriptación	Descifrado
$(1024, 524, 50)$	$67,072$	$32,750$	2^{18}	2^{22}
$(2048, 1608, 40)$	$411,648$	$88,440$	$2^{20,5}$	2^{23}
$(2048, 1278, 70)$	$327,168$	$123,008$	2^{20}	2^{24}
$(2048, 1025, 93)$	$262,400$	$131,072$	2^{20}	$2^{24,5}$
$(4096, 2056, 170)$	$1,052,672$	$524,280$	2^{22}	$2^{26,5}$

Tabla 2.2: Parámetros del esquema de McEliece [19]

2.2. Esquema de Niederreiter

En 1986, Harald Niederreiter propuso la versión dual del criptosistema de McEliece [36]. En un principio empleaba códigos GRS para su construcción pero tras el ataque efectuado por Sidelnikov y Shestakov, que comentaremos en capítulos posteriores, se propuso una versión alternativa del criptosistema que empleaba los códigos de Goppa, al igual que el esquema de McEliece. El esquema de Niederreiter y el de McEliece son equivalentes en el sentido de que mantienen el mismo nivel de seguridad, pero emplean una estructura de clave pública distinta, así como mecanismos de encriptación y decodificación diferentes. A su vez, Niederreiter utiliza la matriz de control del código de Goppa como punto de partida, lo que le permite reducir el tamaño de la clave como comentamos previamente.

Se consideran conocidos los valores de n, k, t mientras que g, P, S se generan de manera aleatoria en secreto. Describiremos a continuación su funcionamiento de manera similar a cómo hicimos con el esquema de McEliece, basándonos de nuevo en [47] y tomando a *Alice* como parte receptora y a *Bob* como parte emisora.

1. Generación de claves: *Alice* generará P_k y S_k de acuerdo con los valores públicos dispuestos para los parámetros.
   a) Selecciona una matriz aleatoria $n \times n$ de permutación P y una matriz S no singular $(n-k) \times (n-k)$.
   b) Toma una matriz de control H para el código de Goppa de dimensión $k = n - mt$ definido por $\Gamma(L, g)$ siendo $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ y g el polinomio de Goppa de grado t sobre $\mathbb{F}_{q^m}$.
   c) Entonces tiene como P_k a la matriz $\hat{H} = S \cdot H \cdot P$ y como $S_k = (P, S, H)$.
2. Encriptación: *Bob* quiere mandar un mensaje m de longitud n y peso t. Es reseñable cómo en este esquema no se añade ningún error de peso t al contrario de lo que sucedía en el McEliece
   a) Calcula el texto cifrado $c = \hat{H} \cdot m^t$ y envía c.
3. Desencriptación: Supongamos que *Alice* ha recibido el texto cifrado $c \in \mathbb{F}^{n-k}$.
   a) Se obtiene $Hz^t = S^{-1}c \in \mathbb{F}^{n-k}$.
   b) Si tomamos el vector z como el vector recibido, interpretándolo como la suma de una palabra del código a la que se le ha añadido el vector error de peso t,

podemos aplicar el algoritmo de Patterson pues está trabajando con un código de Goppa binario. De esa manera obtendrá el vector error $m \cdot P^t$ [4] y al conocer la matriz de permutación P por formar parte de la clave privada, podemos invertirla y recuperar el mensaje m.

Hemos comentado anteriormente que el principal próposito de Niederreiter con este criptosistema era mitigar en cierta medida el problema derivado de los tamaños de clave tan grandes del esquema de McEliece. Tengamos en cuenta que el tamaño de la clave pública viene dado por $(n-k) \times n$, que se corresponde con las dimensiones de la matriz que compone la clave pública, y el de la clave privada por $((n-k)^2+n^2)+(t \times m)+(n \times m)$, sumando las dimensiones de cada una de las matrices que forman la clave privada. Para los parámetros originales del esquema de McEliece, $n = 1024 = 2^{10}$, $t = 50$, $m = 10$ y $k = n - mt = 524$, tenemos para el esquema de Niederreiter claves de $64\,KB$ y $163{,}5\,KB$ respectivamente. Al compararlos con los obtenidos para el caso McEliece, vemos cómo se reducen ciertamente sus tamaños. A pesar de ser equivalentes en cuanto a niveles de seguridad, si se implementan usando los códigos de Goppa, el esquema de Niederreiter ha sido mucho menos investigado que su homólogo, se ha tratado de optimizar en menor medida que éste y ha obtenido un menor análisis de su seguridad. Por todo esto no se ha considerado al esquema de Niederreiter como una posible vía de investigación prometedora como sistema de seguridad postcuántico. Por ello el esquema de McEliece ha sido considerado por delante del de Niederreiter a la hora de seguir desarrollándolo como alternativa criptográfica.

2.3. Equivalencia entre McEliece y Niederreiter

Ya hemos visto que ambos esquemas funcionan de forma similar usando los códigos de Goppa con la pequeña diferencia de que el esquema de Niederreiter posee tamaños de clave ligeramente menores frente a los que ofrece el criptosistema original de McEliece. Cómo ambos son equivalentes si hablamos de seguridad, veamos su equivalencia, o dicho de otra manera, cómo se puede transformar uno en otro y viceversa.

2.3.1. McEliece a Niederreiter

Bob quería enviar un mensaje m de longitud k. Recordemos que en el criptosistema de McEliece disponía para hacerlo de la clave pública, compuesta por la matriz $\hat{G}$ y el entero t, el cual determinaba el peso del vector error $e \in \mathbb{F}_q^n$. Usando estos datos, se tiene el texto cifrado

$$c = m\hat{G} + e.$$

Asumiendo que $\hat{G}$ es la matriz generadora de cierto código obtenemos su respectiva matriz de control $\hat{H}$. Si multiplicamos ahora por $\hat{H}^t$ en la ecuación anterior

$$c\hat{H}^t = m\hat{G}\hat{H}^t + e\hat{H}^t \Longrightarrow c\hat{H}^t = e\hat{H}^t.$$

[4] Por ser P una matriz de permutación y m un vector de peso t, el vector error $m \cdot P^t$ seguirá teniendo peso t, por lo que podremos decodificarlo correctamente.

Así, el lado izquierdo de la ecuación es conocido, pues c es el texto cifrado y $\hat{H}$ es la matriz de control obtenida a partir de la clave pública $\hat{G}$. Se sabe también que el peso del vector error es t. Y sabemos que conociendo el vector error e con el mismo síndrome que el vector recibido c podremos decodificar correctamente. Por tanto, si resolvemos dicha ecuación, hallaremos el vector e, y así tendríamos

$$c - e = m\hat{G}.$$

Si nos fijamos en esta ecuación, nos daremos cuenta de que aún desconociendo la clave privada del esquema de McEliece, simplemente usando su clave pública podríamos recuperar m. Esta equivalencia nos muestra que si el criptosistema de Niederreiter tuviese alguna vulnerabilidad esta también afectaría al esquema de McEliece, pues si conseguimos conocer el vector error e podremos descifrar un mensaje encriptado mediante el McEliece teniendo únicamente su clave pública.

2.3.2. Niederreiter a McEliece

El mensaje y que quería enviar *Bob* tenía longitud n y peso t. El texto cifrado z se obtiene multiplicando por la traspuesta de una matriz $(n-k) \times n$ de la matriz de control pública $\hat{H}$ de cierto código desconocido

$$z = y\hat{H}^t.$$

Empleando una matriz aumentada y álgebra lineal, podemos fácilmente encontrar un vector c de longitud n y peso $\geq t$ de manera que

$$z = c\hat{H}^t \ , \ c = m\hat{G} + y.$$

En estas ecuaciones lo que haríamos es calcular un vector c que tenga el mismo síndrome que y, para lo que tomaríamos c como la suma de una palabra del código más el vector error de menor peso. De hecho, este sistema tiene solución al ser compatible indeterminado, por lo que obtendríamos un vector c de peso $\geq t$ porque sabemos que el error original tiene peso t. Y así, podemos resolver la última igualdad, ya que disponemos del algoritmo de decodificación de McEliece. En este caso, podemos llegar a decodificar una palabra encriptada mediante el esquema de Niederreiter empleando las claves del esquema de McEliece. De nuevo, si este último tuviera alguna vulnerabilidad, también la tendría el de Niederreiter. Es por esto que se garantiza la equivalencia de ambos sistemas en términos de seguridad, siempre que empleen códigos de Goppa idénticos.

2.4. Esquemas alternativos

Venimos recalcando a lo largo de este capítulo que la principal desventaja encontrada al implementar el esquema de McEliece es el tamaño de las claves necesario para garantizar la seguridad requerida. Desde que ha empezado a ganar relevancia se han investigado distintas alternativas para reducir este tamaño. Han aparecido propuestas que emplean distintas familias de códigos como pueden ser los cuasicíclicos o los cuasidíadicos [6], lo

que permitió reducir sustancialmente el tamaño de la clave. El problema de estos códigos es que su estructura algebraica se hace demasiado evidente a través de sus matrices, algo que no sucede en el caso de los códigos de Goppa. Estos ataques se podrían evitar usando códigos que carezcan de una estructura algebraica clara como pueden ser los LDPC o los MDPC presentados anteriormente en la Definición 1.31.

Por la definición que dimos en su momento de estos códigos, podemos apreciar que son bastante similares. Los códigos LDPC, que tienen un peso de fila menor, admiten una matriz de control dispersa así como la implementación de un algoritmo de decodificación eficiente. Sin embargo, eso que les hace atractivos también muestra su debilidad, pues al ser las filas de la matriz de control de peso bajo pueden tomarse como palabras de peso bajo del dual del código. Así, un esquema de McEliece que empleara un código LDPC podría ser atacado buscando dichas palabras a través de su matriz de control y usándolas para reconstruir la matriz de control del código oculto.

Si nos apoyamos en lo estudiado por Misoczki et al. en [35], vieron que bastaba con aumentar moderadamente la longitud y el peso de las filas de la matriz de control para emplear el esquema de McEliece de forma segura. De esa manera se evitan los ataques basados en la recuperación de mensajes así como los basados en la recuperación de la clave. Así, los códigos utilizados serían códigos MDPC. Este aumento de peso trae consigo una disminución en el poder de corrección del protocolo. Sin embargo, este hecho no es tan relevante pues es preferible corregir un menor número de errores con tal de garantizar la integridad de la comunicación. Veremos cómo se construyen estos códigos y la manera de implementarlos en el esquema de McEliece.

2.4.1. Construcción del código MDPC y QC-MDPC

Tengamos en cuenta las Definiciones 1.31 y 1.34.
Para construir un código MDPC aleatorio con parámetros $[n, r, w]$ se escoge una matriz de control aleatoria $H \in \mathbb{F}_2^{r\times n}$ que tenga peso de fila w. Esta matriz tiene una alta probabilidad de poder encontrar en ella un bloque $r \times r$ invertible.
En el caso de su variante cuasicíclica, sabemos que el código tiene longitud $n = n_0 p$ y codimensión r. La matriz de control será de la forma

$$H = [H_0|H_1|\dots|H_{n_0-1}],$$

donde cada H_i es un bloque circulante $p \times p$. La primera fila de esta matriz se define escogiendo un vector aleatorio de longitud n y peso w. Por ser circulante, para obtener el resto de filas basta con ir desplazando cuasicíclicamente esta fila. Esto quiere decir que cada elemento se desplazará uno hacia su derecha, hasta llegar al último elemento del bloque. Cada uno de estos bloques H_i tendrá un peso de fila w_i, teniendo en total $w = \sum_{i=0}^{n_0-1} w_i$. El hecho de que la matriz quede completamente definida por su primera fila es una de las principales causas para emplear la variante cuasicíclica de los códigos MDPC, pues trae consigo un considerable ahorro de memoria. Por cómo está definida H, se puede conseguir de manera sencilla una matriz generadora G. Suponiendo que el

bloque más a la derecha H_{n_0-1} es invertible, construimos la matriz generadora.

$$G = \begin{pmatrix} 1 & 0 & \cdots & 0 & (H_{n_0-1}^{-1} \cdot H_0)^t \\ 0 & 1 & \cdots & 0 & (H_{n_0-1}^{-1} \cdot H_1)^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & (H_{n_0-1}^{-1} \cdot H_{n_0-2})^t \end{pmatrix}$$

2.4.2. Implementación del esquema de McEliece

Vamos a detallar cómo sería el proceso de comunicación en caso de emplear este esquema con un código MDPC (sería análogo en el caso de un código QC-MDPC). Para ello, consideraremos de nuevo la misma estructura que en los ejemplos anteriores.

1. Generación de claves: *Alice* va a generar las claves públicas y privadas correspondientes.
   a) Genera la matriz de control H para un código MDPC capaz de corregir t errores.
   b) A partir de esta matriz H construye su matriz generadora correspondiente G en forma sistemática.
   c) *Alice* publica $P_k = G$ y mantiene en secreto $S_k = H$.
2. Encriptación: En el caso de que *Bob* quiera enviarle un mensaje $m \in \mathbb{F}_2^{n-r}$ hará el siguiente proceso.
   a) Toma un vector error aleatorio $e \in \mathbb{F}_2^n$ con peso t.
   b) Usando la clave pública de *Alice* calcula $c = mG + e$ que será el texto cifrado.
   c) Envía c.
3. Desencriptación: Una vez recibe c, *Alice* recupera m de la siguiente manera.
   a) Calcula mG sirviéndose para ello de un algoritmo de decodificación, el cual emplea la clave privada H en su proceso.
   b) Como G está en forma sistemática, basta con recuperar las $n - r$ primeras posiciones del vector mG y así recuperar el mensaje enviado por *Bob*.

Podemos observar cómo en este caso se suprime el uso de las matrices S y P, además de que al tener la matriz G en forma sistemática se reduce la memoria necesaria. Si se aplican las conversiones de seguridad necesarias [29], sobre las que hablaremos en capítulos posteriores, el proceso se puede llevar a cabo sin traer consigo una disminución en la seguridad.

En el caso de emplear un código MDPC la clave pública tendrá un tamaño de $r \times (n-r)$ bits mientras que si usáramos su variante cuasicíclica su tamaño se vería reducido a $n-r$ bits, teniendo así claves sumamente compactas y ahorrando una cantidad significativa de memoria, sin otorgar además ninguna ventaja adicional a los atacantes. Observando la siguiente tabla y comparándola con la Tabla 2.1.3 podemos ver la significativa reducción en el tamaño de la clave pública del criptosistema, precisamente lo que se buscaba al emplear en la construcción del esquema códigos diferentes a los de Goppa.

Parámetros (n_0, n, r, w, t)	**Tamaño** P_k **(bytes)**	**Nivel de seguridad**
$(2, 9602, 4801, 90, 84)$	4801	80
$(3, 10779, 3593, 153, 53)$	7186	80
$(4, 12316, 3079, 220, 42)$	9237	80
$(2, 19714, 9857, 142, 134)$	9857	128
$(3, 22299, 7433, 243, 85)$	14866	128
$(4, 27212, 6803, 340, 68)$	20409	128
$(2, 65542, 32771, 274, 264)$	32771	256
$(3, 67593, 22531, 465, 167)$	45062	256
$(4, 81932, 20483, 644, 137)$	61449	256

Tabla 2.3: Parámetros del esquema de McEliece con códigos QC-MDPC [35]

2.4.3. Decodificación de los códigos MDPC

Para la decodificación de los códigos MDPC, los autores de [35] decidieron usar una variante del algoritmo *Bit Flipping*[5], en este caso la variante de Gallager [22]. Dicho algoritmo, que es iterativo, permite una capacidad de corrección que aumenta linealmente con la longitud del código pero disminuye al aumentar el peso de las filas de la matriz de control. En el caso considerado, como estaríamos usando un código MDPC, o su variante cuasicíclica, al tener un peso moderadamente mayor que para los códigos LDPC, el algoritmo sufriría una pérdida de rendimiento en cuanto a la capacidad de corrección de errores se refiere.
Veamos cómo funciona el algoritmo *Bit Flipping de Gallager*. Recordemos que al tomar un vector $y = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_2^n$, al multiplicar este vector por la matriz de control H de cierto código, se pueden interpretar las filas de H como un sistema de ecuaciones implícitas que definen dicho código. En ese caso, cada bit no nulo de y tiene asociadas un cierto número de filas de H, denotándose por $M(j)$ el conjunto de ecuaciones asociadas al bit j-ésimo del vector y. Por tanto, un bit no nulo satisfará ciertas ecuaciones, que resultan ser las filas de la matriz de control con valor no nulo en la columna correspondiente a dicho bit.

1. Tomamos el vector recibido $y = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_2^n$ que puede contener errores y calculamos su síndrome $S(y) = yH^t$.
2. Se calcula el número de ecuaciones asociadas a cada bit del mensaje.
3. Estas ecuaciones se distinguen entre satisfechas e insatisfechas. Si el número de ecuaciones insatisfechas asociado a cada bit del mensaje supera un cierto valor umbral T se invierte el valor de dicho bit[6].
4. Se repite el proceso hasta recuperar la palabra del código o hasta que se alcance un número máximo de iteraciones $I_{\text{máx}}$.

[5]El algoritmo *Bit Flipping (BF)* se usa para corregir errores cambiando los bits de los datos recibidos hasta que se obtenga una solución que cumpla con ciertas condiciones de corrección de errores. Se puede encontrar información sobre dicho algoritmo en [32].

[6]El valor de T suele ser una función del número de ecuaciones de control que involucran un bit, $T = \frac{d_j}{2}$ siendo d_j el grado del bit j.

Entrada	Vector recibido y, número máximo de iteraciones $I_{\text{máx}}$
	mientras $iter \leq I_{\text{máx}}$ o $S(y) \neq 0$: 1. Calcular el síndrome del vector y 2. **para** $1 \leq i \leq r$: 2.1. Calcular $s_i = \sum_{j \in N(i)} y_j \ mod\, 2$, donde $N(i)$ es el conjunto de bits relacionados con el i-ésimo bit mediante las ecuaciones de control 3. **para** $1 \leq j \leq n$: 3.1. Calcular $\phi(j) = \sum_{i \in M(j)} s_i$ 3.2. **si** $\phi(j) > T$: $y_j \longleftarrow 1 - y_j$ 4. $iter = iter + 1$
Salida	$y \in \mathcal{C}$ o $\perp$

Tabla 2.4: Algoritmo Bit Flipping de Gallager

Como ya hemos dicho, el incremento en el peso de las filas de la matriz de control conduce a una menor capacidad de corrección de errores y por tanto a tener que ejecutar un mayor número de veces el algoritmo. Para minimizar el problema, en [35] se proponen varias soluciones a la hora de elegir el valor umbral T. Téngase en cuenta por otro lado que al emplear códigos correctores es preferible tener una menor capacidad de corrección mientras que eso se supla con una mayor garantía de comunicación exitosa, es decir, que el mensaje original sea correctamente descifrado.

2.4.4. Esquema para códigos cuasicíclicos

A continuación vamos a presentar un criptosistema de clave pública basado en un cierto código cuasicíclico el cual en este caso no es necesario ocultar. Veremos cómo se construye este esquema usando la métrica de Hamming, pero se podría adaptar también para usarlo con la métrica de rango [55]. A lo largo de esta subsección se utilizará para cada elemento indistintamente su representación como polinomio o su equivalente como vector de sus coeficientes, cuando no haya posibilidad de confusión (1.5).
Se fija un código $\mathcal{C}$, público, y un polinomio aleatorio $h \in \ R = \mathbb{F}_2[x]/(x^n - 1)$, también público. h se corresponde con una matriz $2n \times n$, $\begin{pmatrix} Id \\ H \end{pmatrix}$, siendo la primera columna de H los coeficientes de h y correspondiéndose el resto de columnas con sus permutaciones cíclicas. Téngase en cuenta que el producto $(u, v) \cdot \begin{pmatrix} Id \\ H \end{pmatrix}$ se corresponde en términos de polinomios con $u + v \cdot h$. En general usaremos la representación polinómica, pero el salto a la representación vectorial es inmediato.
A partir del código fijado $\mathcal{C}$ de tipo $[n, k]$ con capacidad de corrección t, y fijando los parámetros w, w_e, w_r con valor entero en el rango de $\frac{\sqrt{n}}{2}$, presentamos el criptosistema de clave pública como antes. Justificaremos al final la elección de los anteriores parámetros.

1. Generación de claves: *Alice* genera el par de claves pública/privada.

a) *Alice* determina el código $\mathcal{C}$ a utilizar y consecuentemente los valores de los parámetros (n, k, t, w, w_e, w_r).
b) Se fija una matriz generadora G y se elige un polinomio aleatorio $h \in R$.
c) *Alice* genera aleatoriamente dos polinomios $y, z \in R$ con $w_H(y) = w_H(z) = w$.
d) Se toma como clave privada $S_k = (y, z)$ y tras calcular el polinomio $s = y + h \cdot z \in R$, se establece como clave pública la tupla $P_k = (G, h, s, t, w_e, w_r)$.

2. Encriptación: *Bob* quiere enviarle un mensaje $m \in \mathbb{F}_2^k$ a *Alice*.
   a) Al conocer la clave pública de *Alice*, genera un polinomio aleatorio $e \in R$ con $w_H(e) = w$, y dos polinomios aleatorios $r_1, r_2 \in R$ con $w_H(r_1) = w_H(r_2) = w_r$.
   b) *Bob* calcula $u = r_1 + h \cdot r_2$ y $v = m \cdot G + sr_2 + e$, siendo ésta última una operación de vectores, por lo que se toman los equivalentes vectoriales de los polinomios calculados anteriormente.
   c) Se envía como mensaje cifrado la pareja $c = (u, v) \in R^2$.

3. Desencriptación: Una vez recibe c, *Alice* recupera m de la siguiente manera.
   a) Usando el polinomio z de la clave privada, realiza la siguiente operación.

$$\begin{aligned} v - u \cdot z &= m \cdot G + sr_2 + e - (r_1 + hr_2)z = \\ &= m \cdot G + (y + h \cdot z)r_2 + e - r_1 z - hr_2 z = \\ &= m \cdot G + (yr_2 - r_1 z + e) \end{aligned}$$

   Nótese que al realizar esta operación, *Alice* ha cambiado el error $s \cdot r_2 + e$ inicialmente añadido a la palabra del código $m \cdot G$ por el error $yr_2 - r_1 z + e$. Puede comprobarse que la elección de los valores de w, w_e, w_r hace que $w_H(s \cdot r_2 + e) > t$, maximizando así la probabilidad[7] de que $w_h(yr_2 + -r_1 z + e) \leq t$ [34].
   b) Como *Bob* añadió un error de peso mayor que la capacidad de corrección del código, no puede ser corregido, sin embargo como *Alice* puede sustituirlo por un error corregible, recupera así m. Esta es la razón por la que el código usado puede hacerse público.

Este criptosistema elimina todos los peligros provenientes de los ataques centrados en conocer cuál es el código usado al poder hacer público el código empleado. En particular, puede utilizarse cualquier código con buenas propiedades y un algoritmo eficiente de decodificación. Como contrapunto de este esquema, cabe la posibilidad de que exista DFR por lo que no podría usarse para el cifrado de mensajes muy importantes. Veremos sin embargo más adelante cómo esto no supone un problema a la hora de usarlo en el diseño de protocolos de intercambio de claves, en particular con un código cuasicíclico.

[7] Que esta probabilidad no sea 1 implica la posibilidad de DFR.

Capítulo 3

Aplicaciones de los esquemas presentados

Gracias a lo visto en los capítulos anteriores hemos podido ir formándonos una idea acerca de la acuciante necesidad de encontrar nuevos sistemas seguros frente a la inminente amenaza de la computación postcuántica. Vimos en la introducción que existen distintos procesos de estandarización, entre ellos el más destacado es el del NIST. Se encuentra en la 4ª y última ronda [50], tras evaluar durante 12 años multitud de propuestas, todas con el propósito de usarse para la criptografía de clave pública y divididas en dos categorías, para cifrado e intercambio de claves y para firma digital.

Varios de los finalistas a elegirse como estándar para el cifrado e intercambio de claves tienen en su base la Teoría de Códigos. El resto de propuestas existentes se basan en isogenias, retículos y criptografía multivariable. Vemos así cómo la criptografía basada en códigos está más vigente que nunca y con visos de seguir aumentando su importancia.

Para consultar los candidatos por el lado de la firma digital en la 3ª ronda, podemos hacerlo en [49]. En este caso no encontramos ningún sistema basado en códigos como propuesta final. Como veremos más adelante, esto se debe a que los tamaños de las firmas son demasiado elevados para construir un esquema práctico. A lo largo de este capítulo estudiaremos alguno de los candidatos además de otras alternativas para ambas categorías, centrándonos en las propuestas basadas en códigos.

3.1. Protocolos de intercambio de claves

Un protocolo de intercambio de claves se usa para construir un mecanismo de encapsulación de claves (MEC en adelante) y tiene como objetivo proteger una clave simétrica haciendo uso de la criptografía asimétrica. De esa manera veremos cómo el esquema de McEliece presentado anteriormente puede ser empleado para construir un MEC. Mostraremos a lo largo de esta sección cómo aplicar el esquema de McEliece [26], el esquema BIKE [1] y más brevemente el esquema HQC [34], que resultan ser todos finalistas de la 4ª ronda del proceso de estandarización del NIST en el apartado de MEC. Debemos tener en cuenta a lo largo de toda esta sección, que los MEC constan en general de tres algoritmos: uno para generar las claves, otro de encapsulación y un último algoritmo de desencapsulación de las mismas. Se denotarán por Keygen, Encap y Decap respectivamente. Existe

la posibilidad de que aunque todas las etapas anteriores se desarrollen correctamente la clave compartida se desencapsule de manera errónea. Está intrínsicamente relacionado con el siguiente concepto.

Definición 3.1. *La Decoding Failure Rate (DFR) se define como la probabilidad de que un intento de descifrado falle, es decir, que se pueda recuperar el mensaje original a partir del texto cifrado recibido, pues éste contiene un número de errores superior a la capacidad de corrección del código usado.*

Sin embargo, este suceso no nos plantearía un mayor problema, pues al ser claves de sesión únicas bastaría con repetir el proceso para tratar de establecer una nueva clave común.

3.1.1. McEliece clásico

En esta subsección, al referirnos a McEliece clásico estamos hablando de la propuesta de MEC para el NIST [26] que se basa en el esquema de McEliece sobre el que hemos hablado a lo largo de todo el trabajo.

Se busca conseguir la creación de una clave criptográfica compartida o simétrica entre dos usuarios mediante el aprovechamiento del cifrado asimétrico. Para lograrlo, el enfoque se centra en el uso de la versión dual del criptosistema de Mcliece, es decir, el esquema de Niederreiter empleando códigos de Goppa binarios. Explicaremos su funcionamiento a continuación, basándonos en los principios estudiados en [47] y [26]. Se tomarán para ello como parámetros: n la longitud del código, k su dimensión, t su capacidad de corrección, q el tamaño del cuerpo y L el conjunto de soporte del código de Goppa.
Como viene siendo habitual, consideraremos como parte receptora a *Alice* y como parte emisora a *Bob*. En este caso quieren establecer una clave de sesión mediante el MEC de McEliece clásico.

- KEYGEN. *Bob* genera el par de claves del McEliece de la siguiente manera.
  1. Elige un polinomio irreducible aleatorio $g(z)$ sobre $\mathbb{F}_{q^m}[z]$ de grado t. Se toma $q = 2$, por lo que tendremos un cuerpo binario. Escoge también el conjunto $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$ de manera aleatoria. Puede determinar así el código de Goppa $\mathcal{C}_\Gamma(L, g(z))$.
  2. Gracias al paso anterior, puede calcular la matriz $\hat{H} = (\hat{h}_{i,j})$ de tamaño $t \times n$. Cada elemento de dicha matriz es $\hat{h}_{i,j} = \alpha_j^{i-1} g(\alpha_i)^{-1}$ para $i \in \{1, 2, \ldots, t\}$ y $j \in \{1, 2, \ldots, n\}$.
  3. Para conseguir una matriz $mt \times n$, escribe cada elemento de $\hat{H}$ como un vector de $\mathbb{F}_q^m$.
  4. Si es posible, haciendo las transformaciones necesarias, se consigue la matriz sistemática $H = (Id_{mt\times mt}|T_{mt\times k})$. En caso de no ser capaces de conseguir dicha matriz, debemos reiniciar el proceso escogiendo un polinomio diferente.
  5. De esta manera, se tienen $P_k = T$ y $S_k = \{g(z), L\}$, siendo P_k la clave pública y S_k la clave privada.
- ENCAP. Una vez que conozca la clave pública T, *Alice* puede empezar el proceso de encapsulación de claves.

1. Genera un vector de peso t, $e \in \mathbb{F}_q^n$, mediante una distribución aleatoria uniforme.
2. Usando la matriz T, *Alice* recupera la matriz H, definida como antes, y emplea el esquema de Niederreiter para calcular el vector $C_0 = He$ de longitud mt.
3. Calcula $C_1 = \mathrm{H}(2, e)$ [1] y crea el texto cifrado $C = (C_0, C_1)$. La longitud del mismo es de $mt + 256$ bits.
4. Establece una clave de sesión $K_s = \mathrm{H}(1, e, C)$ de 256 bits y envía C a *Bob*, que conocerá también su tamaño.

- DECAP. Usando el texto cifrado C que recibe de *Alice*, *Bob* empieza el proceso de desencapsulación para fijar la misma K_s que *Alice*.
  1. Separa el texto cifrado recibido en $C_0 \in \mathbb{F}_q^{mt}$ y $C_1 \in \mathbb{F}_q^{256}$.
  2. Fija $b = 1$ y empieza el proceso de descifrado. Para ello, toma como entradas C_0 y la clave privada $\{g(z), \alpha_1, \alpha_2, \ldots, \alpha_n\}$. Genera también una cadena aleatoria s de n bits.
     a) Calcula un vector $v \in \mathbb{F}_q^n$. Para ello, basta con añadir k ceros al vector C_0.
     b) Gracias al polinomio $g(z)$ y al conjunto de soporte L, *Bob* puede construir el código de Goppa correspondiente como vimos previamente (1.7). De esa manera, usando el esquema de Niederreiter, si es posible encuentra la única palabra c del código tal que $d(v, c) \leq t$. En caso contrario vuelve a empezar el proceso.
     c) Si efectivamente existe la palabra c, toma el vector error e como $e = v + c$. Si su peso es t y $C_0 = He$, se devuelve dicho vector e. En otro caso, se toma $e = s$ y $b = 0$, pues de esta manera posteriormente comprobarán si sus claves de sesión son iguales, y las rechazarán en caso de que no sea así.
     d) Calcula $C_1' = \mathrm{H}(2, e)$. Si no se cumple la igualdad $C_1' = C_1$, se toma $e = s$ y $b = 0$.
     e) Calcula la clave de sesión $K_s' = \mathrm{H}(b, e, C)$.

En caso de que no haya habido ningún fallo durante el proceso de desencapsulación y se cumpla que $C_1' = C_1$, ambas partes tendrán la misma clave de sesión. Lo mismo ocurrirá si *Bob* recibe un vector C válido, pues recuperará correctamente el vector e. En ambos casos, establecerán la misma clave de sesión para el protocolo de clave privada.

Este caso no tiene por qué cumplirse siempre, pues existen ciertos escenarios en los que el proceso puede no ser satisfactorio para alguna de las partes. Puede ocurrir que el texto cifrado que recibe *Bob* haya sido corrompido, y por tanto la clave de sesión que calculará será defectuosa. Para tratar de evitar esto, ambos pueden comunicarse y comparar los valores hash de sus respectivas claves de sesión, es decir comprobar si $\mathrm{H}(1, e, c) = \mathrm{H}(b, e, c)$. Así, en caso de que no tengan los mismos valores, deben empezar de nuevo el proceso para garantizar una posterior comunicación segura.
De los ataques conocidos contra este MEC, sobresale la decodificación de conjuntos de información, sobre la que hablaremos en el siguiente capítulo. Existen otros ataques centrados en la recuperación de las claves, pero han resultado ser mucho más lentos.

[1] H denota la función hash SHAKE256 (Definición 1.36). Los parámetros de entrada en este caso son 0,1 y 2 [26] y se representan como bytes.

Hablemos ahora del tamaño de las claves, que es el principal problema cada vez que hablamos de implementar el esquema de McEliece. En el caso de la clave pública, al estar compuesta únicamente por la matriz T, su tamaño se deriva directamente del tamaño de esta matriz. Por otro lado, el tamaño de la clave privada depende de cada uno de sus componentes. Es decir, del vector $s \in \mathbb{F}_q^n$, el polinomio $g(z)$ de grado t sobre $\mathbb{F}_{q^m}$ y el conjunto $L \subseteq \mathbb{F}_{q^m}$. Teniendo en cuenta que cada fila de la matriz de la clave pública se representa como una cadena de k bits, entonces la clave pública es el resultado de concatenar $mt \cdot k$ de dichas cadenas, luego tiene tamaño $mt \cdot k$. Por su parte, para calcular el tamaño de la clave privada debemos tener en cuenta sus elementos por separado.

- El vector s se almacena como una cadena de n bits.
- Un polinomio mónico irreducible de grado t se representa como una cadena de tm bits, es decir, la concatenación de sus coeficientes $(g_0, g_1, \ldots, g_{t-1})$ como elementos del cuerpo.
- El conjunto L obviamente tiene n elementos distintos de $\mathbb{F}_q$ y cada uno de ellos se representa mediante $\lceil \log_2 q \rceil$ bits.

Dependiendo de la seguridad requerida a la hora implementar este sistema los tamaños de las claves variarán. Se pueden consultar distintos casos en [26].

3.1.2. BIKE

Como ya hemos dicho antes, el mecanismo de encapsulación de claves BIKE sobre el que hablaremos en la sección actual ha participado en el proceso de estandarización del NIST. En un principio sólo había sido considerado como un candidato alternativo hasta la 3ª ronda, y posteriormente pasó a ser uno de los finalistas de la 4ª ronda. Podemos comprobarlo en [49, 50].

Sabemos que BIKE es un MEC. Al igual que el McEliece consta de tres algoritmos, denominados con el mismo nombre que en el caso anterior. Sin embargo, en este caso se construye el esquema de Niederreiter usando los códigos QC-MDPC, expuestos en la Subsección 1.2.3. La seguridad de este esquema se reduce a la variante cuasicíclica del Problema 1.39. El principal propósito de este MEC es el intercambio de claves efímeras, con un par de claves pública/privada para cada sesión, pudiendo utilizar una única vez la clave privada en el proceso de desencapsulación.

A continuación presentaremos las notaciones necesarias para la posterior explicación del funcionamento de este esquema. Para ello nos ayudaremos de lo estudiado en [1] y [37].

1. Parámetros del sistema. Se toma como entrada el nivel de seguridad deseado λ para obtener los valores de los parámetros r, w, t y l.
   - w es el peso de las filas de la matriz de control. Tenemos que w es par y $w/2$ es impar.
   - $t \in \mathbb{Z}^+$ es el peso de Hamming del vector error.
   - $l \in \mathbb{Z}^+$ se refiere al tamaño de la clave simétrica generada.
   - r es la longitud de los bloques circulantes (antes lo denotábamos por p), y como para los códigos QC-MDPC se tenía $n = n_0 p$, tomando $n_0 = 2$ [1],

resulta $n = 2r$. Esta r debe ser lo suficientemente grande para conseguir, en conjunto con w y t, una DFR, vista en la Definición 3.1, lo suficientemente baja para alcanzar el nivel de seguridad λ requerido.

2. Funciones hash. Hablamos anteriormente en la Definición 1.36 sobre lo que son las funciones hash. En este caso, BIKE emplea tres funciones hash distintas H, L y K. Éstas se seleccionan uniformemente al azar y se modelan como oráculos aleatorios. H toma como entrada una cadena de l bits y devuelve una secuencia de $2r = n$ bits con peso t. Las dos restantes funcionan de manera similar teniendo, $\mathrm{L} : \{0,1\}^n \longrightarrow \{0,1\}^l$ y $\mathrm{K} : \{0,1\}^{2l+r} \longrightarrow \{0,1\}^l$.

Una vez conocemos estos conceptos, podemos presentar el mecanismo de encapsulación de claves del esquema, de manera similar a como hicimos en la sección anterior. Para que en este caso, *Alice* y *Bob* establezcan una clave de sesión mediante BIKE, deberán sucederse las siguientes etapas.

- KEYGEN. Toma como entrada los parámetros del sistema presentados. Genera a partir de ellos el par de claves pública/privada.
  1. *Bob* genera los polinomios h_0,h_1 pertenecientes a $\mathcal{R}$, siendo $\mathcal{R}$ el anillo cíclico de polinomios $\mathbb{F}_2[x]/(x^r - 1)$. Ambos polinomios tienen peso $w/2$ y sus coeficientes pueden ser considerados como vectores columna $r \times 1$.
  2. De manera uniformemente aleatoria, escoge un mensaje σ del espacio de mensajes $\mathcal{M} = \{0,1\}^l$.
  3. Así, *Bob* obtiene y envía $P_k = h$, como resultado de calcular $h = h_1 h_0^{-1}$, y mantiene en secreto $S_k = (h_0, h_1, \sigma)$.
- ENCAP. Recibe como entrada la clave pública h compartida por *Bob*.
  1. *Alice* escoge un vector $m \in \mathcal{M}$, de longitud l, siguiendo una distribución aleatoria uniforme.
  2. Calcula $(e_0, e_1) = \mathrm{H}(m)$, siendo e_0 y e_1 vectores de error de longitud r, tales que $w(e_0) + w(e_1) = t$.
  3. Haciendo uso de estos vectores error, *Alice* obtiene $C = (C_0, C_1) = (e_0 + e_1 h, m \oplus \mathrm{L}(e_0, e_1))$ [2].
  4. Envía C a *Bob* y calcula $K_s = \mathrm{K}(m, C)$, mantieniéndola en secreto.
- DECAP. Toma como entrada la clave privada S_k y el texto cifrado C.
  1. *Bob* calcula el síndrome $S = C_0 h_0$.
  2. Usando el decodificador BGF, sobre el que hablaremos más adelante, decodifica S y así obtiene los vectores error e'_0 y e'_1. Si $w(e'_0) + w(e'_1) \neq t$ o la decodificación no es exitosa, se devuelve $\perp$ y se detiene.
  3. En caso contrario, *Bob* calcula $m' = C_1 \oplus \mathrm{L}(e'_0, e'_1)$. Si $\mathrm{H}(m') \neq (e'_0, e'_1)$, entonces toma $m' = \sigma$.
  4. Por último, establece $K_s = \mathrm{K}(m', C)$.

Gracias al proceso que acabamos de presentar, ambas partes podrán establecer una clave de sesión K_s segura. Si el receptor legítimo, en este caso *Bob*, recibe un texto cifrado C válido, siempre podrá calcular K_s a partir de dicho texto. Por otro lado, cualquier atacante

[2] $\oplus$ denota la operación lógica XOR de dos bits.

que intente recuperar K_s desconociendo la clave privada verá sus intentos frustrados. Esto se debe al hecho de que la decodificación de $C_0 h_0$ para un código cuasicíclico arbitrario no es factible, al ser este problema NP-completo. Además, el receptor lícito puede cotejar la integridad de C_0 calculando $\mathrm{H}(m') = (e'_0, e'_1)$, pues es indispensable para recuperar la clave común correctamente.

Para facilitar al lector la comprensión del esquema presentado, vamos a profundizar un poco más en las etapas de encapsulación y desencapsulación de claves, pues alguna de las afirmaciones hechas puede no ser trivial a simple vista. Tendremos en cuenta la equivalencia entre vectores y polinomios presentada anteriormente en la Subsección 2.4.4.

- Cuando en la etapa de encapsulación hablamos de obtener un texto cifrado, en particular C_0, vemos cómo se multipilica el vector error e_1 por la clave pública h. Recordemos que en realidad esta clave pública se obtiene a partir de los polinomios h_0 y h_1. Éstos son los que definen el código QC-MDPC que se emplea. Sabemos que la matriz de control de este código es una matriz circulante de bloques. De hecho se puede representar como
$$\hat{H} = \begin{pmatrix} Id_{r\times r} \\ H_1 H_0^{-1} \end{pmatrix}.$$
Los bloques H_0 y H_1, al ser circulantes, quedan completamente definidos por sus primeras columnas, pues el resto de las columnas son permutaciones cíclicas de las primeras, que en este caso resultan ser h_0 y h_1 respectivamente. Es por esto que se publica la matriz correspondiente al polinomio $h = h_1 h_0^{-1}$, y no las correspondientes a los polinomios h_0 y h_1. Por tanto tenemos
$$C_0 = \begin{pmatrix} e_0 & e_1 \end{pmatrix} \cdot \hat{H} = e_0 + e_1 H_1 H_0^{-1} \equiv e_0 + e_1 h_1 h_0^{-1} = e_0 + e_1 h,$$
precisamente como hemos visto antes. Debemos tener en cuenta que la parte izquierda de la equivalencia está tratando la multiplicación de vectores y matrices, mientras que la parte derecha se trata de operaciones con los polinomios que tienen como coeficientes los componentes de dichos vectores.
- Por otro lado, en la etapa de desencapsulado de las claves, es necesario recuperar (e_0, e_1) valiéndonos para ello del texto cifrado C_0. En este caso, se toma como matriz de control del código cuasicíclico $H = \begin{pmatrix} H_0 \\ H_1 \end{pmatrix}$, que es fácil ver que es matriz de control del mismo código que H^t, y como matriz generadora $G = \begin{pmatrix} H_1 & H_0 \end{pmatrix}$, pues sobre $\mathbb{F}_2$ se comprueba fácilmente que $G \cdot H = 0$. Esta igualdad se cumple ya que al ser bloques circulantes entonces $H_0 H_1 = H_1 H_0$, pues dichas matrices por los polinomios cuyos coeficientes forman la primera columna y sabemos que el producto de polinomios es conmutativo. Supongamos que tenemos un vector $v = (v_1, v_2, \ldots, v_r)$, por lo que su vector codificado correspondiente será $v \cdot G = \begin{pmatrix} vH_1 & vH_0 \end{pmatrix}$. Si se asume el error de peso t se puede representar como $e = \begin{pmatrix} e_0 & e_1 \end{pmatrix}$, entonces
$$r = v \cdot G + e = \begin{pmatrix} vH_1 + e_0 & vH_0 + e_1 \end{pmatrix}.$$
Calculemos ahora el síndrome del vector recibido, teniendo en cuenta que la suma

de la traspuesta de dos matrices es igual a la traspuesta de la suma de dos matrices.

$$S = r \cdot H = \begin{pmatrix} vH_1 + e_0 & vH_0 + e_1 \end{pmatrix} \cdot \begin{pmatrix} H_0 \\ H_1 \end{pmatrix} = (vH_1 + e_0) \cdot H_0 + (vH_0 + e_1) \cdot H_1 =$$
$$= vH_1H_0 + e_0H_0 + vH_0H_1 + e_1H_1 = e_0H_0 + e_1H_1 = \begin{pmatrix} e_0 & e_1 \end{pmatrix} \cdot H$$

Esta igualdad se verifica por la conmutatividad de los bloques circulantes explicada anteriormente. La igualdad también se cumple debido a que trabajamos sobre un cuerpo de característica 2. Es fácil de ver, si tomamos la equivalencia como polinomios, $C_0h_0 = (e_0 + e_1h)h_0 = (e_0 + e_1h_1h_0^{-1})h_0 = e_0h_0 + e_1h_1 \equiv e_0H_0 + e_1H_1 = \begin{pmatrix} e_0 & e_1 \end{pmatrix} \cdot H$. Teniendo en cuenta lo comentado en el punto anterior, el resultado de calcular C_0h_0 va a ser equivalente a calcular el síndrome S.

Así, para recuperar e_0 y e_1, se usa un decodificador basado en el algoritmo *BF* y que comparte ciertos detalles con el algoritmo *BF de Gallager* visto antes en la Subsección 2.4.3. Como el algoritmo *BF* puede traer consigo una DFR más alta, se usa una variante algo más compleja, conocida como *decodificador Black-Gray-Flip (BGF)*, que ha sido el recomendado para ser usado en la etapa de desencapsulación de BIKE al conseguir minimizar la complejidad y la DFR. La matriz H será la entrada que tome el algoritmo para llevar a cabo la decodificación. Recordemos que dicha matriz queda completamente definida por h_0 y h_1. Para consultar los detalles de dicho algoritmo se podrá hacer en [1, 37], ya que nos abstendremos de ahondar en este asunto en el contexto de nuestra investigación actual, ya que excede los límites de nuestro alcance. Podemos encontrar también en [1, Tabla 5, Tabla 6] los tamaños de las claves para los distintos niveles de seguridad requeridos.

3.1.3. HQC

Hablaremos del último MEC propuesto como candidato en el concurso de estandarización del NIST que está basado en códigos. Es muy similar a los MEC presentados en las anteriores secciones, por lo que no repetiremos los elementos en común en esta sección, con la excepción de las funciones de cifrado y descifrado propias del esquema HQC presentado en la Subsección 2.4.4. Lo hemos incluido para así abarcar las tres propuestas basadas en códigos del NIST. En este caso, el MEC usa de forma combinada un código $\mathcal{C}$ de tipo $[n, k]$ sobre $\mathbb{F}_q$ capaz de corregir hasta t errores y otro código cuasicíclico de tipo $[2n, n]$ doble circulante, es decir, cuya matriz de control está definida por dos bloques circulantes. Otra de las diferencias es que en este caso ambos códigos son públicos.
Considerando las funciones hash G, H y K y los parámetros (n, k, t, w, w_e, w_r) definidos anteriormente, desarrollaremos su funcionamiento a continuación.

- KEYGEN. Toma como entrada los parámetros del sistema presentados. Genera a partir de ellos el par de claves pública/privada.
  En este caso, lleva a cabo el mismo proceso que la generación de claves de la Subsección 2.4.4.
- ENCAP. Recibe como entrada la clave pública P_k compartida por *Bob*.
  1. Se genera un elemento $m \in \mathbb{F}_2^k$ y $b \in \mathbb{F}_2^{128}$ aleatorios.
  2. Se toma la dupla (h, s) a partir de los valores públicos disponibles.

3. Usando la función hash G, se obtiene $\theta = \mathrm{G}(m, h, b)$.
4. Se genera el texto cifrado $c = (u, v)$ a partir del mensaje plano m y el error θ. Usando la función hash K se obtiene la clave de sesión $K_s = \mathrm{K}(m, c)$.
5. Se calcula $d = \mathrm{H}(m)$. Se envía a *Bob* la terna (c, d, b).

- DECAP. Toma como entrada la terna compartida por *Alice*.
  1. Descifra el texto cifrado recibido de *Alice* c, para así obtener m'.
  2. Calcula $\theta' = \mathrm{G}(m', h, b)$. Realiza un cifrado a partir del mensaje plano m' y el error θ' para obtener c'.
  3. Si $c \neq c'$ o $d \neq \mathrm{H}(m')$, se interrumpe la comunicación. En caso contrario, se establece $K_s = \mathrm{K}(m, c)$ como clave de sesión.

Este MEC está fuertemente basado en el criptosistema de clave pública presentado en la Subsección 2.4.4. Se puede consultar en mayor detalle en [34, 55].

3.1.4. Otros protocolos

A lo largo de esta sección hemos estudiado tres de los candidatos en el proceso de estandarización del NIST para el intercambio de claves, todos ellos basados en códigos. Tras un largo estudio, finalmente en el 2022 fueron publicados los candidatos que iban a ser estandarizados en este proceso [51], y aquellos que iban a seguir siendo estudiados [48], entre los que se encuentran los tres MEC que hemos tratado en esta sección. Existen otras propuestas interesantes que hemos investigado pero no se incluyen en detalle en este trabajo por tener menor relevancia. Aún así, las citaremos a continuación.

- Una de ellas, es el protocolo de intercambio de claves CAKE. Éste basa de nuevo su implementación en el criptosistema de McEliece usando un código QC-MDPC. Permite el uso de claves efímeras, venciendo a su vez un ataque que apareció contra los códigos MDPC y ofrece un procedimiento de generación de claves altamente eficiente para los criptosistemas basados en códigos QC-MDPC.
- Otra de las propuestas investigadas ha sido el protocolo de intercambio de claves Ouroboros que agrupa las propiedades de los protocolos MDPC-McEliece junto con los protocolos HQC. Esto le permite ofrecer una gran simplicidad a la hora de decodificar y alcanzar con parámetros más pequeños el mismo nivel de seguridad. Esto se debe a la estructura cíclica doble en la que se basa. Sin embargo, no fue considerado como candidato al NIST debido a su escasa madurez y falta de comprobaciones al respecto, pero es interesante al ser otra propuesta dentro de la criptografía basada en códigos.

Podemos encontrar más información sobre ellos en [4] y [16] respectivamente.

3.2. Firma digital

Una vez que hemos estudiado distintas alternativas existentes relativas al cifrado e intercambio de claves, nos vamos a centrar en esta sección en las propuestas de firma digital basadas en códigos. La firma digital es el proceso por el cual se intenta evitar la suplantación de la identidad digital [15]. Actualmente, las firmas digitales más extendidas

son aquellas basadas en el algoritmo RSA o en el protocolo de ElGamal, como el DSA. Últimamente se están desarrollando varias firmas basadas en curvas elípticas, como la que se usa en *Whatsapp* o *Bitcoin*. El problema con estos protocolos ya existentes es de nuevo la aparición de la computación postcuántica, que puede poner en jaque a estos sistemas, tan necesarios para nuestra vida cotidiana.
Es por eso, que desde el NIST, conjuntamente con el proceso de estandarización para los protocolos de intercambio de claves, se abrió el proceso para la estandarización de una firma digital que pudiera ser utilizada en el escenario postcuántico. Se puede consultar en [51] los candidatos seleccionados para su estandarización. Debido al excesivo tamaño de las claves, ninguno de estos candidatos se basa en la Teoría de Códigos, estando dichas propuestas basadas en retículos, funciones hash e incluso en redes neuronales recurrentes. Esto contrasta con la predominancia de los candidatos basados en códigos en la parte del cifrado e intercambio de claves. Sin embargo, desde un punto de vista teórico sí resulta interesante estudiar protocolos de firma digital basados en Teoría de Códigos, entre ellos el que presentaremos a continuación.

3.2.1. Firma CFS (Courtois, Finiasz y Sendrier)

Como hemos mencionado anteriormente, el RSA es uno de los protocolos de firma más utilizados hoy en día. Esto no deja de ser curioso, pues el esquema de McEliece y el RSA empezaron a ser estudiados en los años 70 del siglo pasado. Sin embargo, debido al gran tamaño de las claves que genera el McEliece, éste resultó ser menos exitoso y por tanto recibió un menor interés. Ha sido a partir de la aparición de ataques exitosos contra el RSA que gracias al algoritmo de Shor y al aumento de la capacidad de computación han podido explotar las debilidades de estos criptosistemas, cuando se han empezado a buscar esquemas de firmas digital basados en McEliece. Esto se debe a que, como ya sabemos, todos los ataques conocidos hasta la fecha han resultado necesitar un tiempo exponencial para llevarse a cabo, al estar basado en un problema NP-completo.
Sin embargo, hasta hace no demasiado se creía que el esquema de McEliece no podía ser empleado como base de una firma digital. Sólo se contemplaban los esquemas de pruebas de conocimiento cero[3]. Pero la idea de usar dichas pruebas resultó infructuosa porque, a pesar de ofrecer una seguridad excelente, las firmas tenían un tamaño demasiado grande, por lo que no se pudo implementar ningún esquema que fuera práctico. Para vencer estos contratiempos, se han estudiado distintas maneras para conseguir un esquema de firma digital basado en códigos. Lo veremos a continuación ayudándonos de lo estudiado en [14].

Debemos saber que cualquier función de puerta trampa, vimos este concepto en la Definición 1.35, como por ejemplo el logaritmo discreto en el que se basa ElGammal, permite implementar firmas digitales aprovechando la capacidad única del propietario de la clave pública para invertir la función. Sólo se podrá usar para firmar mensajes cuyo valor hash esté en el espacio de los textos cifrados. Es por ello, que dicho esquema deberá alcanzar la decodificación completa. Vamos a ver cómo han conseguido esto en el caso de los códigos de Goppa.
Para conseguir una firma digital eficiente nos basamos en el esquema de McEliece, con su

[3]Una prueba de conocimiento cero es un método criptográfico en el que una de las partes puede demostrar a la otra que conoce un valor específico, como puede ser una contraseña o una clave secreta, sin revelar ninguna información adicional sobre ese valor.

configuración presentada anteriormente en la Subsección 2.1.1. Recordemos que se tenía como clave pública P_k la matriz $\hat{G} = S \cdot G \cdot P$, siendo G una matriz generadora del código de Goppa oculto $\mathcal{C}$. Y como clave privada S_k, teníamos el código binario de Goppa $\mathcal{C}$ con una capacidad de corrección t, la matriz invertible S y la matriz de permutación P. Teniendo esto en cuenta, la firma se genera de la siguiente manera.

1. Dado un documento $\mathbf{D}$ y una cierta función hash H, conocida públicamente, con una salida de longitud n, se calcula el valor hash de dicho documento, H($\mathbf{D}$).
2. Usando el algoritmo de decodificación, conocido por el receptor al saber exactamente cuál es el código empleado, decodifica el vector H($\mathbf{D}$) para hallar el correspondiente vector de información $\mathbf{F}$.
3. Se establece $\mathbf{F}$ como la firma del documento $\mathbf{D}$. La firma será válida siempre que se emplee correctamente la clave privada, pues posteriormente será necesaria la clave pública a la hora de verificarla.

Para verificar la firma recibida se siguen los pasos enumerados a continuación.

1. Tomando la firma $\mathbf{F}$, se codifica de acuerdo al esquema de McEliece y el vector resultante actuaría en este caso como palabra del código.
2. Se compara ahora el vector cifrado obtenido con el valor hash del documento H($\mathbf{D}$).
3. Si ambos vectores se encuentran a una distancia menor o igual que la capacidad de corrección del código, es decir, menor o igual que el valor conocido t, se asume que el único que ha podido encontrar una palabra del código tan cercana es quién tiene la clave privada, entonces sabe decodificar y por tanto la firma será válida. En caso contrario, se rechazará esta firma.

Una de las principales desventajas que se encontraban al tratar de implementar un esquema de firma digital basado en códigos era la complejidad de calcular dicha firma [25]. Especialmente porque en el paso 2 del anterior algoritmo, al emplear un algoritmo de decodificación, concretamente el correspondiente al código usado, y considerar una palabra aleatoria de longitud n, normalmente se obtiene una palabra con un error de peso $> t$. Por lo tanto, la capacidad de corrección del código se vería sobrepasada. Una de las soluciones que buscaron los autores de [14] fue ser capaces de implementar un algoritmo que consiguiera la denominada decodificación completa.

Como se ha mencionado en los preliminares, la decodificación completa se centra en hallar la palabra del código más cercana a cualquier palabra recibida aunque esta palabra supere la capacidad correctora del código empleado. Se consigue así poder decodificar más allá de la capacidad de corrección de dicho código. Si pensamos en las palabras del código como los centros de bolas de radio $\leq t$, entonces en el caso de la decodificación completa estas bolas no serán disjuntas. Denotaremos por $\delta_{\text{mín}}$ al entero más pequeño para el cual el volumen de una esfera de radio $t + \delta_{\text{mín}}$ es $> 2^{n-k}$, es decir, existen más vectores dentro de las bolas que en todo $\mathbb{F}^n$, por lo que algunos vectores estarán contenidos en varias bolas distintas. $\delta_{\text{mín}}$ se calcula como mín$\{\delta \in \mathbb{N} | \sum_{i=0}^{t+\delta} \binom{n}{i} > 2^{n-k}\}$. No se trata de una decodificación, pues la palabra obtenida con este algoritmo no es necesariamente la más cercana al vector, dado que este puede estar en las bolas correspondientes a varias palabras del código y cómo se devuelve una cualquiera puede no ser la más cercana. Sin embargo esto no será un inconveniente para el uso que se le va a dar. Este algoritmo procede de la siguiente manera.

1. Se empieza intentando conseguir corregir $t+1$ errores en vez de los t errores originales.
2. Para ello, se cambia el valor de un bit cualquiera y se trata de decodificar el vector resultante mediante un algoritmo de decodificación única.
3. Si se consigue decodificar, esto significa que el bit cambiado era uno de los $t+1$ incorrectos. Si no, este bit no estaría entre los erróneos por lo que se probaría con un nuevo bit.
4. Análogamente, cambiando bits de δ en δ, se pueden llegar a corregir hasta $t + \delta_{\text{mín}}$ errores.

El problema es que la decodificación aumenta tanto su coste computacional como el número de vectores devueltos por el algoritmo de manera exponencial con el valor de δ, por lo que si aumenta demasiado δ tendríamos un problema computacional. De hecho, si tenemos una palabra de longitud n que contiene $t+\delta$ errores y queremos corregirla, nuestra probabilidad de tener éxito a la hora de decodificar δ errores más allá de la capacidad de corrección del código será la de elegir δ coordenadas aleatorias del vector y que esas efectivamente sean parte de las $t+\delta$ coordenadas erróneas es decir

$$\mathbb{P}_{\text{éxito}} = \frac{\binom{t+\delta}{\delta}}{\binom{n}{\delta}},$$

donde el numerador expresa el número de combinaciones posbiles de elegir δ errores en $t+\delta$ posiciones, mientras que el denominador describe el número de combinaciones posibles de elegir δ errores en n posiciones. El valor de δ debe ser pequeño y tal que se pueda decodificar cualquier palabra a una distancia $< t + \delta_{\text{mín}}$, excepto un número despreciable de ellas, lo que se conoce como decodificación casi completa.

Recordemos que los parámetros del esquema de McEliece original eran $n = 1024$, $k = 524$ y $t = 50$. Para estos valores, se obtiene $\delta_{\text{mín}} = 61$, conllevando a una probabilidad de éxito en la decodificación $\approx 2^{-222}$, lo cual no es aceptable. Por este motivo, en [14] tras estudiar el rendimiento de la firma para distintos parámetros, se determinaron como parámetros del código de Goppa a emplear $n = 65536$ y $k = 65392$, pudiendo así corregir $t = 9$ errores con $\delta_{\text{mín}} = 2$. Teniendo en cuenta estos valores y basándonos en el algoritmo de firma digital presentado previamente, conseguiríamos una firma digital basada en el esquema de McEliece de la siguiente forma. Veamos en primer lugar cómo se genera la firma.

1. Calculamos el valor hash del documento y obtenemos una palabra de n bits de longitud.
2. Usando el algoritmo de decodificación casi completa, vamos cambiando $\delta_{\text{mín}} = 2$ bits aleatorios de la palabra de n bits hasta encontrar una palabra del código a distancia $t + \delta_{\text{mín}} = 11$ del valor hash obtenido.
3. Conseguimos el mensaje de longitud k correspondiente a dicha palabra del código, y lo empleamos como firma.

Y ahora, presentaremos el proceso de validación de dicha firma.

1. Codificamos la palabra de k bits usando la clave pública.

2. Calculamos el hash del documento y comparamos ambos vectores.
3. La firma será legítima si ambas palabras difieren como mucho en $t + \delta_{\text{mín}} = 11$ bits.

La seguridad conseguida con este esquema es buena, pero la firma es de 65392 bits, un tamaño demasiado grande. Para tratar de reducir la envergadura de la firma y conseguir un esquema de firma practicable, se encontraron distintas alternativas basadas en la misma idea y que presentaremos a continuación.

Con el objetivo de reducir el tamaño de la firma siempre en mente, se propuso una variante del esquema anterior haciendo uso en este caso de la versión dual del McEliece, el esquema de Niederreiter, presentado en la Sección 2.2. Recordemos que en este caso la clave pública era la matriz $\hat{H} = S \cdot H \cdot P$ siendo H la matriz de control del código $\mathcal{C}$. Este código $\mathcal{C}$ con capacidad de corrección t junto con la matriz no singular S y la matriz de permutación P componían la clave privada. El problema de decodificar se convierte ahora en hallar, a partir del síndrome recibido, la palabra con peso mínimo que tenga dicho síndrome. Además, en este caso el hash del documento obtenido tiene ahora una longitud de $n - k$ bits. La principal ventaja proviene de que en el caso anterior había que decodificar una palabra que podía o no estar a una distancia corregible por el código, y en este caso el síndrome corresponde a una palabra dentro de la capacidad de corrección del código. Veamos entonces cómo se calcula la firma.

1. Se calcula el valor hash del documento $\mathbf{D}$.
2. El vector obtenido $\mathrm{H}(\mathbf{D})$ de $n - k$ bits es el síndrome de cierto vector.
3. Usando el algoritmo de decodificación de Niederreiter para $\mathrm{H}(\mathbf{D})$ obtenemos el vector $\mathbf{F}$ correspondiente a dicho síndrome que se establece como la firma del documento.

Por tanto, para hallar el valor de la firma basta con calcular el error correspondiente al síndrome, que es el hash de $n - k$ bits obtenido. Esto sustituye la decodificación de una palabra aleatoria de longitud n. Como en general en nuestro caso se cumple que $n - k < n$, conseguiremos un aumento en la velocidad y una reducción en el tamaño del hash, la longitud de la firma y el tamaño de la clave pública. Y para verificar esta firma el proceso es el siguiente.

1. Dada la firma $\mathbf{F}$ se codifica siguiendo el esquema de Niederreiter.
2. Se compara el valor obtenido con el valor hash del documento.
3. En caso de que ambos sean iguales, la firma será válida. Mientras que se rechazará si no se cumple la igualdad.

Al construir en este caso la firma usando el esquema de Niederreiter, tomaremos como firma una palabra de peso $t + \delta_{\text{mín}}$, fijando en este caso un peso 11, y longitud 65536. Recordemos que para firmar un documento basta con decodificar el síndrome obtenido al calcular el hash de dicho documento. Esta firma se envía comprimida, para lo que podemos emplear dos métodos.

- Escribimos los índices de los 11 bits no nulos de la palabra. Como tiene 2^{16} bits, tendremos entonces $11 \times 16 = 176$ bits.

- Numeramos todas las palabras de peso 11 y utilizamos el número x correspondiente como firma. La palabra con 1's en las posiciones $i_1 < i_2 < \ldots < i_t$ se codifica como

$$x = \binom{n - i_1}{t} + \binom{n - i_2}{t - 1} + \ldots + \binom{n - i_t}{1} + 1.$$

  Así, se tiene una firma de longitud $\lceil \log_2 \binom{65536}{11} \rceil = 151$ bits.

Podemos ver cómo se ha reducido significativamente el tamaño de la firma respecto de la obtenida en el caso de instanciar el esquema de McEliece. Sin embargo, el hecho de utilizar firmas tan cortas abre la posibilidad de que estos sistemas sean atacados independientemente de la fortaleza de la función de puerta trasera usada. A pesar de esto, los ataques no representan una amenaza real contra nuestro esquema de firma, pues la memoria necesaria para llevarlos a cabo ronda los $2^{72} \times 72$ bits, o lo que es lo mismo, más de 10^{11} TB. Los autores de [14] propusieron en este mismo trabajo un par más de esquemas de firma, utilizando pequeñas modificaciones, que no consideramos necesarias de explicación al ser similares a lo visto, llegando a conseguir firmas con una longitud de 111 bits con un tiempo de verificación inferior al segundo.

Todas estas propuestas han demostrado ser seguras, sin embargo la falta de estandarización en comparación con otros esquemas ampliamente usados como el DSA o RSA y el coste computacional que conllevan las han relegado a un segundo plano, y, en consecuencia, a no ser consideradas en el proceso del NIST. Adicionalmente, los autores del trabajo [14] aportan como casos de prueba las siguientes tablas en las que comparan distintos parámetros de las 3 variantes de firma presentadas en su trabajo y el protocolo original con otros esquemas conocidos como RSA o ElGamal.

Criptosistema base	RSA	ElGamal	Curva Elíptica	McEliece
Protocolo de firma	RSA	DSA	ECDSA	CFS
Tamaño de los datos	1024	160/1024	160	65536
Mejor ataque estructural	2^{102}	2^{102}	Desconocido	2^{149}
Longitud de firma	1024	320	162	65392
Tamaño P_k (KB)	0,2	0,1	0,1	1152
Tiempo de firma (ms)	9	1,5	5	10000
Tiempo de verificación (ms)	9	2	6	32

Tabla 3.1: Comparación de la firma CFS con otros esquemas de firma conocidos [14]

Si nos fijamos en esta primera tabla, podemos ver cómo la firma CFS, si nos centramos en la seguridad, alcanza el mayor nivel de todas las expuestas. Sin embargo, podemos ver también cómo necesita cantidades de memoria mucho mayores que el resto, ya sea para almacenar claves o debido a la longitud de la firma y, además, sus tiempos de firma y verificación son más grandes que el resto. Por tanto, aunque es más segura, la ganancia no es tan significativa como para tomarla como estándar al tener en cuenta el resto de características.

Firma CFS	Original	Variante A	Variante B	Variante C
Tamaño de los datos	65536	144	144	144
Longitud de firma	65392	151	150	111
Tamaño P_k (KB)	1152	1152	1152	1152
Tiempo de firma (s)	10	10	10	10
Tiempo de verificación (ms)	32	$< 0{,}001$	$< 0{,}001$	1000

Tabla 3.2: Comparación de las distintas variantes de firma CFS [14]

Y en esta segunda tabla, en la que se comparan las distintas variantes presentadas por los autores en [14] para la firma CFS, vemos cómo se consigue minimizar el impacto de algunos de estos problemas. Sin embargo, aunque algunas variantes consiguen unas longitudes de firma y una necesidad de memoria menores, la mayoría siguen teniendo tiempos de firma demasiado altos y los tamaños de la clave pública son demasiado grandes.

Hemos podido ver a lo largo de esta sección cómo implementar de manera eficiente un esquema de firma digital basado en códigos. La firma CFS ha demostrado ser la única propuesta exitosa basada en códigos hasta el momento pues ha conseguido tamaños aceptables para su aplicación práctica. En nuestro estudio hemos encontrado otras alternativas como el esquema de identificación de Stern [53], el esquema propuesto en [25] que, basándose en el McEliece, ha conseguido tiempos de ejecución menores que la firma CFS o la mejora sobre el esquema de identificación de Giraut presentado en [5], apoyado en el problema NP-completo de subcódigos equivalentes. Debido a falta de espacio y una menor relevancia hemos aportado las citas bibliográficas en las que se pueden consultar dichas propuestas.

Capítulo 4

Ataques y seguridad

A lo largo de todo este trabajo hemos insistido en la misma idea: la computación cuántica supone una peligrosa amenaza para los criptosistemas que son seguros actualmente. De hecho, en [9] se afirma que cuando existan los computadores cuánticos, el RSA, el DSA de curvas elípticas y otros muchos criptosistemas serán rotos por completo usando el algoritmo de Shor. Hay ciertos criptosistemas que usan una función de un sentido con puerta trasera como los esquemas de firma digital basados en hash o el criptosistema de clave pública, sobradamente conocido en este punto por nosotros, el esquema de McEliece que sin embargo han demostrado resistir estos ataques. A pesar de ello, no debemos pensar que esto mantiene los criptosistemas basados en códigos a salvo de todo peligro. Por ejemplo, el algoritmo cuántico de Grover reduce en un factor $\mathcal{O}(\sqrt{N})$ la complejidad de los ataques, lo que implicaría la necesidad doblar el tamaño de las claves de los criptosistemas para garantizar el mismo nivel de seguridad.

Para comprender por qué los ordenadores cuánticos suponen tal desafío para la criptografía moderna, me gustaría ilustrar brevemente cómo funciona la computación cuántica para tener así una mejor comprensión de las mejoras que ésta trae consigo. La principal característica que marca la diferencia entre un ordenador clásico y uno cuántico es la manera que tienen ambos de realizar los cálculos. Mientras que el ordenador tradicional en cada punto del tiempo se encuentra en un único estado, el computador cuántico puede estar en varios estados a la vez en el mismo momento, lo cual se conoce como superposición cuántica [2]. Esto lo podemos trasladar a las unidades de memoria en las que guardan la información. Un ordenador tradicional normalmente utiliza bits para guardar la información, que pueden tener únicamente el estado 0 o 1 en cada momento. Por su parte, los ordenadores cuánticos emplean qubits, o bits cuánticos, que pueden estar en superposición de los dos estados base, $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ y $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, cada uno de ellos con una cierta probabilidad. Como no tienen un valor concreto en cada momento, dicho valor se determina de la siguiente manera. Una vez que midamos el valor del qubit, esto llevará a su estado a colapsar en uno de sus estados base [38] con una cierta probabilidad. Es por ello, que n qubits pueden estar en superposición hasta en 2^n estados al mismo tiempo, lo que supone una de las causas del significativo aumento de la capacidad de cómputo frente a los ordenadores clásicos.

Tras ofrecer una visión general muy introductoria de los principios en los que se basa la computación cuántica, veremos a continuación algunos conceptos relativos a la seguridad de los criptosistemas. Posteriormente hablaremos acerca de ciertos ataques existentes contra la criptografía basada en códigos, pues es el tema que nos incumbe en este trabajo. Y por último, veremos también algunas maneras de contrarrestarlos.

4.1. Nociones de seguridad

La mayoría de los esquemas de encriptación buscan conseguir la indistinguibilidad de los textos cifrados. Si un criptosistema es indistinguible entonces el adversario no será capaz de distinguir pares de textos cifrados donde uno de ellos sea el correspondiente al mensaje original. Es decir, ningún adversario podría distiniguir los textos cifrados con una probabilidad mayor que $\frac{1}{2} + \epsilon$, siendo ϵ una función despreciable dependiente del parámetro de seguridad λ. Los siguientes conceptos que vamos a presentar aparecen de forma generalizada al hablar sobre la seguridad de los criptosistemas.

Se plantea un desafío en el que el adversario, denotado generalmente como *Eve*, genera dos mensajes de la misma longitud. Por su parte, *Alice* debe encriptar al azar uno de ellos. Y *Eve* una vez los reciba trata de adivinar cual de los dos mensajes ha sido encriptado.

Definición 4.1. *Un criptosistema capaz de resistir el desafío presentado se dice que posee indistinguibilidad bajo ataques de texto plano elegido (IND-CPA).*

El siguiente nivel de seguridad busca lo mismo que el juego anterior siguiendo un proceso muy similar. En este caso, *Eve* posee una herramienta adicional, puede usar un oráculo de encriptación o desencriptación[1], es decir, puede encriptar o desencriptar mensajes antes de obtener el texto cifrado de *Alice*. Se puede entender como un acceso temporal a los algoritmos usados por *Bob*.

Definición 4.2. *Un criptosistema resistente al juego presentado se dice que tiene indistinguibilidad bajo ataques de texto cifrado elegido no adaptativos (IND-CCA1).*

Por último, tenemos la indistinguibilidad que todos los criptosistemas aspiran a conseguir, pues es el nivel de seguridad más alto. En este caso, *Eve* tiene acceso a los oráculos también tras recibir el texto cifrado de *Alice*, pero no puede usar el oráculo de desencriptación con el texto cifrado que acaba de recibir. O lo que es lo mismo, puede encriptar y desencriptar tantos textos cifrados como desee para tratar de averiguar cuál de los dos mensajes ha sido elegido por *Alice*.

Definición 4.3. *Se define como indistinguibilidad bajo ataques de texto cifrado elegido adaptativos (IND-CCA2) la propiedad de los criptosistemas que resulten inmunes a este último caso.*

Para estudiar estos conceptos nos hemos ayudado de lo visto en [23].

[1]En el campo de la criptografía, el término oráculo denota una entidad o sistema capaz de proporcionar respuestas a consultas particulares con confiabilidad y uniformidad. Este concepto sirve como marco teórico para evaluar la solidez de los protocolos criptográficos.

4.2. Ataques contra la CBC

El hecho de hacer tanto hincapié en conseguir un cierto nivel de seguridad en nuestros criptosistemas se debe a que existen terceras partes implicadas en el proceso, con fines que no siempre son benignos. Es por eso que debemos ser capaces de proteger correctamente la información transmitida y ser capaces de corregir los errores que puedan ser introducidos en dicha información, para lo que hacemos uso de los criptosistemas basados en códigos.

En lo relativo a la criptografía, existen multitud de ataques que intentan franquear la seguridad impuesta en los criptosistemas. Nosotros en este trabajo nos hemos querido centrar en los relativos a la criptografía basada en códigos. Podemos encontrar un listado de algunos de ellos en [2, Sección 4.6]. Veremos a lo largo de esta sección varios de los ataques que hemos estudiado, los cuales actúan contra los criptosistemas basados en códigos que hemos analizado previamente en este trabajo.

4.2.1. Ataque de decodificación de conjuntos de información

Este ataque es más conocido por sus siglas en inglés ISD, por lo que así lo denotaremos a partir de ahora siempre que hablemos de él.
Cuando hablamos acerca de ataques contra la criptografía basada en códigos, nos encontramos con que el ataque ISD es el más eficiente a la hora de emplearlo contra el esquema de McEliece, y por consiguiente, contra todos los criptosistemas basados en dicho esquema. Este ataque se centra en resolver el problema general de decodificación, es decir, tomando como entrada el texto cifrado c devolver el mensaje original m, o equivalentemente el vector error e. Lo presentaremos a continuación de forma general, ayudándonos de lo estudiado en [47], para posteriormente ver ciertas mejoras que se han presentado para aumentar su eficiencia computacional.

Sea $\hat{G}$ la matriz pública del criptosistema de McEliece, que junto con el entero positivo t compone la clave pública. Recordemos que dicha matriz es el resultado de multiplicar la matriz invertible S, la matriz generadora del código subyacente G y la matriz de permutación P, constituyendo estas tres matrices la clave privada. Por tanto, si queremos cifrar un mensaje m, tomamos el vector error $e \in \mathbb{F}_2^n$ de peso t.

$$\begin{aligned} m \cdot \hat{G} + e =& (m_1, m_2, \ldots, m_n) \cdot (\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_n) + (e_1, e_2, \ldots, e_n) = \\ =& (m_1 \cdot \hat{G}_1, m_2 \cdot \hat{G}_2, \ldots, m_n \cdot \hat{G}_n) + (e_1, e_2, \ldots, e_n) = \\ =& (m_1 \cdot \hat{G}_1 + e_1, m_2 \cdot \hat{G}_2 + e_2, \ldots, m_n \cdot \hat{G}_n + e_n) = \\ =& (c_1, c_2, \ldots, c_n) = c \in \mathbb{F}_2^n \end{aligned}$$

siendo c el texto cifrado resultante, y denotando $\hat{G}_i$ con $i \in \{1, 2, \ldots, n\}$ a la i-ésima columna de la matriz $\hat{G}$.
El aspecto crucial identificado en este sistema y lo que le hace vulnerable ante este ataque es el hecho de que el peso del vector error es significativamente pequeño en relación a la longitud del código. Esto quiere decir que solo t de las n coordenadas del vector error son no nulas. Por ello, si el atacante fuera capaz de averiguar k de las $n - t$ coordenadas

restantes, formando las k columnas correspondientes de $\hat{G}$ un menor no nulo, lo cual no es siempre posible, y de c que correspondan con el valor 0 en dicha coordenada de e, entonces la restricción de esas k columnas de c y la matriz conocida $\hat{G}$, se podría tomar para formar un sistema $\bar{c} = m \cdot \bar{\hat{G}}$.
Para ejemplificarlo, tomaremos las k coordenadas $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, n\}$ tales que para cada $1 \leq j \leq k$ se tenga $e_{i_j} = 0$. Este conjunto de coordenadas es lo que se conoce como conjunto de información, y es lo que da su nombre al ataque. Sirviéndonos de la restricción que acabamos de comentar, tendríamos

$$\underbrace{(c_{i_1}, c_{i_2}, \ldots, c_{i_k})}_{\bar{c}} = m \cdot \underbrace{(\hat{G}_{i_1}, \hat{G}_{i_2}, \ldots, \hat{G}_{i_k})}_{\bar{\hat{G}}}$$

Considerando esta ecuación, si $\bar{\hat{G}}$ es invertible, el atacante ya podría recuperar el mensaje simplemente multiplicando el texto cifrado interceptado c por la inversa de la submatriz cuadrada. En [47], se demuestra que el coste computacional de este algoritmo es demasiado elevado tomando los parámetros originales de McEliece, $n = 1024$, $k = 524$ y $t = 50$. Esto hace que el algoritmo que estamos empleando no sea el apropiado en términos de eficiencia. El problema con este ataque, es que aunque resulte satisfactorio sólo permite recuperar el mensaje enviado, por lo que la clave sigue permaneciendo oculta para posteriores ataques.

Este ataque se puede utilizar también como base de otros algoritmos. Encontramos por ejemplo el algoritmo de Lee-Brickell [30] que trata de encontrar el vector error contenido en el mensaje recibido. Sirviéndose del ataque ISD consigue atacar el criptosistema de una manera más eficiente. Al igual que al presentar otros algoritmos en este trabajo, vamos a presentar de una manera simple el algoritmo, para luego ver su implementación de forma esquemática. Se toma la matriz generadora G del código $\mathcal{C}$ de tipo $[n, k]$, un texto cifrado $y \in \mathbb{F}_2^n$ y el parámetro del algoritmo $p \in \mathbb{N}$ que veremos a continuación para qué sirve.

1. Se toma un conjunto de información, definido como antes, cualquiera I de tamaño k.
2. Se toma la submatriz de G correspondiente al conjunto de información seleccionado, que se denota como G_I.
   a) Si existe la inversa de esta matriz, se toman las coordenadas de y correspondientes al conjunto de información para obtener el vector y_I y se calcula $G' = G_I^{-1} \cdot G$.
   b) Si no existe la inversa de G_I, volvemos al primer paso.
3. Se calcula el vector $y' = y - y_I \cdot G_I$.
4. Se usa el parámetro p como el tamaño de subconjuntos de I, se calcula para cada subconjunto $\{a_1, a_2, \ldots, a_p\} \subseteq I$, y cada $x_1, x_2, \ldots, x_p \in \mathbb{F}_q \setminus \{0\}$ el vector $\hat{g} = \sum_{i=1}^{p} x_i \cdot G'_{a_i}$.
5. Se toma como vector error $e = y - \hat{g}$
   a) Si el peso de dicho vector error es t, se habrá encontrado el error contenido. Se devuelve y se puede decodificar el texto cifrado y.
   b) En caso de que $w_H(e) \neq t$, entonces se vuelve al paso 1.

Una vez entendido el funcionamiento del algoritmo, podemos presentarlo de una manera más formal.

Entrada	Matriz generadora G, texto cifrado $y \in \mathbb{F}_2^n$, parámetro $p \in \mathbb{N}$
	1. Elegir un conjunto de información cualquiera I de tamaño k 2. **si** $\exists\ G_I^{-1}$: Calcular y_I, G_I escogiendo las columnas correspondientes de G y hallar $G' = G_I^{-1} \cdot G$ **si no**: Volver al paso 1. 3. Calcular $y' = y - y_I \cdot G_I$ 4. Para cada subconjunto $\{a_1, a_2, \ldots, a_p\} \subset I$ y cada $x_1, x_2, \ldots, x_p \in \mathbb{F}_q \setminus \{0\}$ calcular $\hat{g} = \sum_{i=1}^{p} x_i \cdot G'_{a_i}$ 5. Tomar como $e = y' - \hat{g}$. 6. **si** $w_H(e) = t$: Devolver el vector error e **si no**: Volver al paso 1.
Salida	Vector error e de peso t

Tabla 4.1: Algoritmo de Lee-Brickell

Haciendo uso de este algoritmo, se consigue una considerable mejora en el coste computacional del ataque para los parámetros originales de McEliece, comparándolo con el coste computacional previo a aplicar este algoritmo [47].

Otra de las variantes que se ha estudiado para aumentar la eficiencia de este ataque es el uso del algoritmo de Stern [52], que explicaremos a continuación. En este caso, la idea sobre la que se fundamenta el algoritmo es diferente a lo explicado anteriormente, pues se basa en el problema de encontrar una palabra del código cuyo peso sea bajo. Busca dicha palabra ya que si se tiene conocimiento de ella se puede decodificar un código lineal y se consigue así romper el criptosistema de McEliece.

Veamos cómo se puede decodificar encontrando una palabra del código de peso bajo. Sea un código $\mathcal{C}$ de longitud n y distancia mínima d sobre $\mathbb{F}_2$ y un elemento $y \in \mathbb{F}_2^n$ que se encuentra a una distancia w de una palabra c de dicho código, $w_H(e) = w$. Se cumplirá que $e = y - c$ es un elemento de $\mathcal{C} + \{0, y\}$, que al ser la suma de dos subespacios, resulta ser otro subespacio que define un código lineal. Si $d > w$, entonces el elemento $e \in \mathcal{C} + \{0, y\}$ de peso w no puede estar[2] en el código $\mathcal{C} \subset \mathcal{C} + \{0, y\}$, de donde se deduce que forma parte de $\mathcal{C} + \{y\}$. Aplicando esto a nuestro caso, donde el código empleado $\mathcal{C}$ sabemos que tiene $d \geq 2t + 1$, resulta que un texto cifrado $y \in \mathbb{F}_2^n$ se encuentra a distancia t de una única palabra del código $c \in \mathcal{C}$. El atacante tiene conocimiento de la matriz generadora $\hat{G}$ de $\mathcal{C}$, pues es parte de la clave pública. Así, si añade el vector y al conjunto de generadores de $\mathcal{C}$, al no pertenecer al código, pues es una palabra a distancia w entonces es linealmente

[2]Pues se sabe que la distancia mínima de un código lineal $\mathcal{C}$, d, es igual a su peso mínimo, por lo que no podrá haber una palabra del código con peso menor que el peso mínimo del código.

independiente de las filas de $\hat{G}$, puede construir una matriz generadora del código lineal $\mathcal{C} + \{0, y\}$. Sabiendo que la única palabra del código $\mathcal{C} + \{0, y\}$ con peso t es $e = y - c$, si mediante cierto algoritmo se pueden encontrar las palabras de peso más bajo del código $\mathcal{C} + \{0, y\}$, entonces se puede encontrar dicha palabra e. Usando esta palabra, obtendría $c = y - e$, que es una palabra del código con matriz generadora $\hat{G}$. Y por ser esta matriz pública, puede recuperar a partir de ella el texto plano correspondiente. Esta idea también se puede emplear contra el esquema de Niederreiter realizando las modificaciones necesarias.

Para tratar de optimizar el funcionamiento del ataque ISD los autores realizaron ciertas mejoras sobre el algoritmo de Stern. Explicaremos dichas mejoras a continuación para posteriormente presentar esquemáticamente dicho algoritmo.

El paso más costoso en la idea que hemos presentado se centra en el momento de encontrar la palabra del código $\mathcal{C} + \{0, y\}$ de peso w. Es en este paso en el que han concentrado sus esfuerzos los autores de [12], y sobre lo que profundizaremos a continuación. Para entender la manera en la que se busca la palabra del código de peso w diferenciaremos las siguientes etapas. Se parte de una matriz de control H de un código dado. Tengamos en cuenta que la matriz de control H la tomamos generalmente en forma vertical, pero en este caso concreto por simplicidad vamos a tomar la matriz de control en forma horizontal H^t. Debemos también tener en mente que las palabras del código se corresponden con combinaciones lineales nulas no triviales de las columnas de H^t. Esta afirmación es generalmente conocida y se puede encontrar en [54].

1. Normalización de la submatriz identidad. Empleando operaciones elementales de filas, se consigue una matriz identidad $(n-k) \times (n-k)$ de entre las n columnas de H^t, pues sabemos que la matriz de control de un código se puede escribir siempre en forma sistemática.
2. Elección de conjuntos. Cada una de las columnas de esta submatriz Id se corresponde ahora con la fila que tenga un 1 en la columna de la submatriz. Teniendo ésto en cuenta, se escogen aleatoriamente un número l de columnas de la matriz identidad resultante de la etapa anterior que van a formar el subconjunto Z. Así, el subconjunto Z se corresponde con un conjunto de l filas de H^t. Las k columnas restantes se reparten de manera independiente y uniforme para crear otros dos conjuntos X e Y adicionales.
3. Cálculo de vectores de colisión. Ahora, dentro de los conjuntos X e Y vamos a tomar subconjuntos de tamaño p. Así, para cada subconjunto A de X de dicho tamaño, se calcula la suma mód 2 de las columnas de A para cada una de las l filas correspondientes a Z. De ese modo, se obtiene el vector $\pi(A)$ de longitud l. De manera análoga, se obtiene $\pi(B)$, siendo B un subconjunto de tamaño p de Y.
4. Búsqueda de colisiones y construcción de la palabra del código. Para cada colisión $\pi(A) = \pi(B)$, se calcula la suma mód 2 de las $2p$ columnas completas de $A \cup B$. Resulta un vector de $n-k$ bits, que tendrá l bits nulos debido a la colisión. Por tanto, el peso del vector resultante vendrá dado por la suma de los bits de esas $2p$ columnas que no se encuentren en las l filas determinadas por Z. Si ese peso fuese $w - 2p$ sumando $w - 2p$ columnas de la submatriz identidad de H^t, se obtiene el vector 0. Así esas $w - 2p$ columnas, junto con las $2p$ columnas que componen A y B forman una combinación lineal de w columnas de H^t y como se mencionó anteriormente, esto determinaría la palabra del código de peso w.

De esta manera, el algoritmo de Stern permite averiguar una palabra del código de peso bajo, lo cual como hemos dicho antes permitiría romper el esquema de McEliece. En el artículo [12] se estudian ciertos avances que permiten modificar este algoritmo y conseguir así mejorar su eficiencia, como por ejemplo permitir múltiples elecciones del conjunto Z entre otras. Esto supondrá que si empleamos el algoritmo de Stern para atacar el criptosistema, considerando los parámetros originales del McEliece, un atacante es capaz de decodificar 50 errores con un coste computacional menor que el de los ataques ISD anteriores [3], una mejora muy significativa si lo comparamos con factores de trabajo previos. Sin embargo, el coste computacional del ataque sigue siendo elevado, por lo que no se emplea contra los criptosistemas basados en códigos estudiados.

Entrada	Matriz de control H, de tamaño $n \times (n-k)\times$, de un código $[n,k]$ sobre $\mathbb{F}_2$, entero positivo w, parámetros del algoritmo l y p
	1. Seleccionar $n-k$ de las n columnas de H^t 2. Seleccionar un subconjunto aleatorio Z de tamaño l de esas $n-k$ columnas 3. Dividir las k columnas restantes en dos conjuntos X e Y 4. Buscar subconjuntos $A \subseteq X$, $B \subseteq Y$ con $\pi(A) = \pi(B)$ tal que $\sum_{c \in A} c + \sum_{\bar{c} \in B} \bar{c}$ tenga peso $w - 2p$ 5. Si no existen tales palabras en el código, volver al paso 1.
Salida	Palabra del código de peso w

Tabla 4.2: Algoritmo de Stern

Hemos visto como el ataque ISD puede ser implementado contra criptosistemas basados en el esquema de McEliece. Podemos consultar por ejemplo el ataque existente contra el MEC presentado anteriormente en 3.1.1 en [47]. Además, al ser el ataque más efectivo contra la criptografía basada en códigos, se están estudiando diversas mejoras para hacer el ataque efectivo en la práctica, como la variante de Leon o Canteaut & Chabaud, sobre las que se puede recabar más información en [19], y que no se incluyen al ser similares a la variante de Stern presentada en este trabajo. Este ataque ISD a priori hace vulnerable nuestros criptosistemas basados en códigos, sin embargo debemos tener en cuenta que a pesar de ser eficiente no es práctico si el código es lo suficientemente largo.

4.2.2. Otros ataques

Hemos detallado algunos de los ataques contra la criptografía basada en códigos. No deja de crecer el número de ataques debido a la relevancia actual y futura de la CBC. La parte positiva es que la existencia de dichos ataques conlleva, un proceso iterativo de mejora de los futuros criptosistemas. Adicionalmente a los ataques sobre los que hemos hablado a lo largo de esta sección, podemos encontrar también los siguientes.

- El ataque de reenvío de mensajes o de mensajes relacionados se basa en la premisa de que un mensaje sea cifrado dos veces con la misma clave, aunque un mismo mensaje produce un mensaje cifrado diferente en cada ocasión. Sin embargo, si se

[3] Se puede consultar los parámetros a utilizar y su estudio en [12].

da esta condición, el atacante podría comparar los mensajes cifrados interceptados para recuperar el mensaje original. Consigue un factor de trabajo mucho menor contra el esquema de McEliece que el ataque ISD analizado al principio. Gracias a lo estudiado en [47] y [23], y por los motivos que acabamos de ver, el esquema de McEliece no es IND-CCA2 seguro, pues no resiste los ataques adaptativos bajo textos cifrados elegidos, que se han presentado previamente en la Definición 4.3. Veremos sin embargo más adelante maneras de conseguir dicha seguridad para nuestros criptosistemas. Para evitar el ataque basta con usar una clave pública diferente para cada cifrado.

- El ataque de Sidelnikov-Shestakov [46] fue capaz de romper en 1992 el esquema de Niederreiter que empleaba códigos GRS en su construcción al recuperar su clave secreta. A raíz de este suceso, el esquema de Niederreiter se vió obligado a usar los códigos de Goppa que evitaban este ataque, pues eliminaban la estructura algebraica concreta de la matriz de control de los códigos GRS, que fue la falla explotada por este ataque. Nos abstendremos de dar más detalles sobre el ataque en particular debido a la necesidad de ampliarlo más allá del alcance de este estudio. Si se tiene interés en profundizar más acerca de dicho ataque se puede consultar en [19, 39].
- La variante cuántica del algoritmo ISD, que aparece en [9] y trae consigo un aumento en la velocidad a la que se puede realizar dicho ataque, debido a la mayor capacidad de la computación cuántica, como explicamos en la introducción de este capítulo.
- El algoritmo de división de soporte explicado en [19], aprovecha la estructura de los códigos de Goppa para recuperar los vectores error introducidos.
- La decodificación uno entre muchos (DOOM) presentada en [45], en la que el atacante tiene acceso a un gran número de textos cifrados, y se conforma con ser capaz de decodificar uno de ellos. Esto permite reducir de manera significativa la complejidad del ataque.
- Los ataques de reacción, que se pueden consultar por ejemplo en [19] o [38], y se centran en observar la reacción del receptor legítimo al recibir mensajes previamente interceptados y alterados por el atacante. Son una variante de los ataques de textos cifrados elegidos adaptativos, relacionados con la Definición4.3.
- Los ataques basados en un distinguidor de códigos de Goppa con tasas de información altas [20] han surgido como una nueza amenaza. Éstos centran sus esfuerzos en recuperar la clave del criptosistema mediante un sistema lineal de ecuaciones polinómicas. Una vez resuelto permite distinguir la matriz de un código de Goppa de la de un código aleatorio, un problema que era considerado en principio NP-completo para la familia de códigos de Goppa. Aunque no es viable en la práctica, pone de relieve la necesidad de seguir investigando sobre la seguridad de los códigos de Goppa pues puede que éstos sean más vulnerables de lo que parecían en un principio.

4.3. Medidas de seguridad

En la sección anterior hemos podido ver algunos de los ataques existentes cuando hablamos de los criptosistemas relativos a la criptografía basada en códigos. Aunque sigue siendo una de las mejores candidatas en el escenario de la criptografía postcuántica, debemos tener en cuenta las debilidades expuestas anteriormente para conseguir sistemas lo

más seguros posibles. A lo largo de esta sección, veremos distintas maneras de incrementar la seguridad de los criptosistemas sobre los que hemos hablado durante todo el trabajo.

Como hemos podido comprobar, el esquema de McEliece se postula como un fuerte candidato en la seguridad postcuántica. Es por ello que veremos a continuación algunas pequeñas modificaciones que se han propuesto a la hora de implementar este criptosistema para conseguir un nivel de seguridad más alto.

- La primera idea que se nos ocurre a la hora de incrementar la seguridad del McEliece es la de aumentar la longitud del código usado. En [12] recomiendan emplear valores de n que sean potencias de 2, ya que así el tamaño de la clave pública del esquema se optimiza mejor. Esto conlleva a un cierto incremento en el tiempo de decodificación, sin embargo no existe a priori ningún contratiempo a la hora de generar las claves sobre un cuerpo $\mathbb{F}_{2^d}$ de tamaño muy superior a n.
- Se está estudiando la opción de usar un algoritmo de *list decoding* aplicable a los códigos de Goppa binarios irreducibles, que como sabemos son los que se emplean en el criptosistema de McEliece. Previamente mencionamos que este algoritmo permitía decodificar un número de errores mayor a la capacidad de corrección del código, en este caso concreto este número sería aproximadamente $n - \sqrt{n(n-2t-2)} \geq t+1$. De esta manera, se podría introducir un número más alto de errores a la hora de encriptar los mensajes, lo que complicaría más aún la tarea de decodificación al atacante[4].

Existen distintas propuestas en [12] para los parámetros del esquema con sus respectivos niveles de seguridad alcanzados.
A continuación veremos ciertas transformaciones realizadas sobre el esquema de McEliece para alcanzar un mayor nivel de seguridad.

4.3.1. Conversiones de seguridad

Para abordar estas amenazas y mejorar la robustez de estos esquemas, se han desarrollado varias conversiones de seguridad. Éstas buscan aumentar la fortaleza de los esquemas originales, proporcionando protecciones adicionales contra ataques específicos y mejorando la seguridad general del sistema criptográfico. En esta subsección, vamos a hablar sobre algunas de ellas, en específico las aplicadas al esquema de McEliece.

Kobara e *Imai* estudiaron las conversiones de *Pointcheval* y *Fujisaki-Okamoto* y señalaron que una vez aplicadas al esquema de McEliece se conseguía la seguridad IND-CCA2 deseada [42]. La única desventaja era la redundancia que se añadía de manera innecesaria y que ellos han tratado de reducir con varias propuestas [29]. Para estudiar las distintas conversiones en mayor profundidad, podemos consultar [19]. Vamos a presentar la primera de ellas, pues las restantes funcionan de un modo parecido y no aportan mayor relevancia si explicamos una de ellas, pues la conversión de *Fujisaki-Okamoto* [21] es muy similar a la de *Pointcheval* mientras que *Kobara* e *Imai* [29] han desarrollado mejoras de éstas.

Necesitaremos introducir las siguientes notaciones, compartidas por las conversiones

[4]Aunque existe la posibilidad de que el *list decoding* devuelva más de una palabra para un texto cifrado, si el esquema de McEliece es CCA2-seguro es trivial distinguir cuál es la correcta[12].

mencionadas, para poder presentar correctamente la conversión de *Pointcheval*. Se tomarán r, r' como números aleatorios, H por una función hash con salida de longitud $\log_2 \binom{n}{t}$ bits, R como un generador pseudoaleatorio de números, y $\mathcal{E}$ y $\mathcal{D}$ como la función de encriptación y desencriptación respectivamente del esquema. $Conv$ será una función de conversión que calcula un vector error e de tamaño n y peso t a partir de un vector dado, y $MSB_n(m)$ y $LSB_n(m)$ indican los n bits más a la derecha y a la izquierda de m respectivamente.

Conversión de Pointcheval [41]. Una función $f : X \times Y \longrightarrow Z$ es parcialmente de puerta trasera unidireccional (*Partially Trapdoor One-Way Function*) si no es posible recuperar $x \in X$ o $y \in Y$ simplemente a partir de su imagen $z \in Z$, pero el conocimiento de cierto secreto permite una inversión parcial. Trasladando esto al esquema de McEliece, tenemos que basa su seguridad en la asunción de que su función de encriptación $\mathcal{E}$ es una PTOWF. Esto es se debe a que toma como entrada el mensaje m y el vector error e, y puede ser invertida para recuperar m si y sólo si se conoce la matriz del código Gopppa subyacente. *Pointcheval* demostró que cualquier función PTOWF puede ser usada en un criptosistema para alcanzar la seguridad IND-CCA2. Veamos ahora las transformaciones realizadas sobre los algoritmos de encriptación y desencriptación del esquema.

Entrada	Números aleatorios r y r', mensaje m, función de encriptación $\mathcal{E}$
	1. $z = \mathrm{H}(m\|r)$, donde $\|$ denota la concatenación de vectores. 2. $z = Conv(z)$ 3. $c_0 = \mathcal{E}(r', z)$ 4. $c_1 = \mathrm{R}(r') \oplus (m\|r)$ 5. $c = (c_0\|c_1)$
Salida	Texto cifrado c de McEliece

Tabla 4.3: Conversión de Pointcheval (Encriptación)

Entrada	Texto cifrado c de McEliece, función de desencriptación $\mathcal{D}$
	1. $c_0 = MSB_n(c)$ 2. $c_1 = LSB_{Len(m)+Len(r)}(c)$ 3. $(r', z) = \mathcal{D}(c_0)$ 4. $(m\|r) = c_1 \oplus \mathrm{R}(r')$ 5. **si** $c_0 = \mathcal{E}(r', Conv(\mathrm{H}(m\|r)))$: Devolver m **si no**: Rechazar c
Salida	Mensaje m

Tabla 4.4: Conversión de Pointcheval (Desencriptación)

Conclusiones

A lo largo de este trabajo hemos investigado la criptografía basada en códigos, enfocándonos principalmente en el esquema de McEliece, al ser la propuesta más segura dentro de este campo cuando hablamos de implementar un sistema seguro contra los ataques cuánticos. Hemos visto la manera de emplear estos esquemas junto con las distintas variantes existentes que buscan reducir el tamaño de las claves generadas manteniendo niveles de seguridad similares. Para ello, hemos visto cómo se pueden aprovechar ciertas propiedades de los códigos usados en su construcción. Hemos desarrollado también distintos algoritmos de decodificación, analizando su eficiencia, y las aplicaciones de los esquemas presentados a través de las propuestas de estandarización del NIST. Por último, hemos revisado algunos de los ataques que amenazan ciertos esquemas presentados y hemos repasado algunas medidas que ayudan a contrarrestarlos.

Me parece conveniente resaltar el estudio que hemos hecho acerca de las propuestas de estandarización centrándonos en la base teórica que éstas contienen y que nos ha permitido ver la importancia de las matemáticas en nuestro mundo. En esta memoria hemos podido observar también cómo los códigos correctores de errores se han convertido gracias a sus propiedades en una de las alternativas más fuertes en el escenario de la criptografía postcuántica. Debemos resaltar sobre el resto los códigos de Goppa que proporcionan una base sólida a la hora de implementar sistemas criptográficos seguros.

Sabemos que la criptografía basada en códigos evoluciona constantemente debido entre otras cosas al rápido avance de la tecnología existente. Por ello existen varias vías de investigación diferentes como la optimización de los algoritmos de decodificación, la construcción de códigos más eficientes o la implementación práctica de criptosistemas basados en códigos en los dispositivos inteligentes usados en el día a día. Consideré en su momento estudiar la criptografía aplicada a la tecnología blockchain, aunque acabé descartando esta idea al tener mayor peso tecnológico que matemático. Sin embargo hay que resaltar como en su construcción emplean multitud de conceptos matemáticos como las curvas elípticas, la Teoría de Grafos o las funciones hash para garantizar la integridad de la información compartida y una correcta comunicación.

Durante estos cuatro meses este trabajo me ha permitido ver una aplicación práctica de las matemáticas en el mundo real algo que no siempre es fácil. Me ha proporcionado una comprensión más profunda de la criptografía basada en códigos y he comprendido también el desafío existente de equilibrar seguridad y eficiencia. Ha potenciado también el desarrollo de habilidades críticas y capacidad de decisión.

En resumen, hemos podido ver el uso de la Teoría de Códigos como base de algunas de las tecnologías que se están investigando a día de hoy para ser la base de la seguridad criptográfica futura. Esto ha nos ha permitido de nuevo poner de relieve el papel fundamental de las matemáticas en nuestro mundo.

Bibliografía

[1] Nicolas Aragon et al. «BIKE: bit flipping key encapsulation». En: (2022).

[2] Chithralekha Balamurugan et al. «Post-Quantum and Code-Based Cryptography—Some Prospective Research Directions». En: *Cryptography* 5.4 (20 de dic. de 2021), pág. 38. ISSN: 2410-387X. DOI: 10.3390/cryptography5040038. URL: https://www.mdpi.com/2410-387X/5/4/38.

[3] Sasha Barg. «Some new NP-complete coding problems». En: *Problemy Peredachi Informatsii* 30.3 (1994), págs. 23-28.

[4] Paulo SLM Barreto et al. «CAKE: C ode-Based A lgorithm for K ey E ncapsulation». En: *Cryptography and Coding: 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings 16*. Springer. 2017, págs. 207-226.

[5] Thierry P Berger, Cheikh Thiécoumba Gueye y Jean Belo Klamti. «A NP-complete problem in coding theory with application to code based cryptography». En: *International Conference on Codes, Cryptology, and Information Security*. Springer. 2017, págs. 230-237.

[6] Thierry P Berger et al. «Reducing key length of the McEliece cryptosystem». En: *International Conference on Cryptology in Africa*. Springer. 2009, págs. 77-97.

[7] Elwyn Berlekamp, Robert McEliece y Henk Van Tilborg. «On the inherent intractability of certain coding problems (corresp.)» En: *IEEE Transactions on Information theory* 24.3 (1978), págs. 384-386.

[8] Elwyn R Berlekamp. *Algebraic coding theory (revised edition)*. World Scientific, 2015.

[9] Daniel J Bernstein. «Grover vs. mceliece». En: *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*. Springer. 2010, págs. 73-80.

[10] Daniel J Bernstein. «List decoding for binary Goppa codes». En: *International Conference on Coding and Cryptology*. Springer. 2011, págs. 62-80.

[11] Daniel J Bernstein y Tanja Lange. «Post-quantum cryptography». En: *Nature* 549.7671 (2017), págs. 188-194.

[12] Daniel J Bernstein, Tanja Lange y Christiane Peters. «Attacking and defending the McEliece cryptosystem». En: *Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA, October 17-19, 2008 Proceedings 2*. Springer. 2008, págs. 31-46.

[13] Richard E Blahut. *Algebraic codes for data transmission*. Cambridge university press, 2003.

[14] Nicolas T Courtois, Matthieu Finiasz y Nicolas Sendrier. «How to achieve a McEliece-based digital signature scheme». En: *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*. Springer. 2001, págs. 157-174.

[15] José Ignacio Iglesias Curto. *Notas de Teoría de Códigos*. Apuntes de la asignatura. Códigos y Criptografía. Salamanca, España, 2023.

[16] Jean-Christophe Deneuville, Philippe Gaborit y Gilles Zémor. «Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory». En: *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*. Springer. 2017, págs. 18-34.

[17] Whitfield Diffie y Martin E Hellman. «New directions in cryptography». En: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022, págs. 365-390.

[18] Said El Hajji, Abderrahmane Nitaj y El Mamoun Souidi. *Codes, Cryptology and Information Security*. Springer, 2017, págs. 230-237.

[19] Daniela Engelbert, Raphael Overbeck y Arthur Schmidt. «A summary of McEliece-type cryptosystems and their security». En: *Journal of Mathematical Cryptology* 1.2 (2007), págs. 151-199.

[20] Jean-Charles Faugere et al. «A distinguisher for high-rate McEliece cryptosystems». En: *IEEE Transactions on Information Theory* 59.10 (2013), págs. 6830-6844.

[21] Eiichiro Fujisaki y Tatsuaki Okamoto. «Secure Integration of Asymmetric and Symmetric Encryption Schemes». En: *Advances in Cryptology — CRYPTO '99*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, págs. 537-554. DOI: 10.1007/3-540-48405-1_34.

[22] Robert Gallager. «Low-density parity-check codes». En: *IRE Transactions on information theory* 8.1 (1962), págs. 21-28.

[23] Nolive Gnan. «OVERVIEW OF THE MCELIECE CRYPTOSYSTEM AND ITS SECURITY». En: ().

[24] Valerii Denisovich Goppa. «A new class of linear correcting codes». En: *Problemy Peredachi Informatsii* 6.3 (1970), págs. 24-30.

[25] Farshid Haidary Makoui, Thomas Aaron Gulliver y Mohammad Dakhilalian. «A new code-based digital signature based on the McEliece cryptosystem». En: *IET Communications* 17.10 (2023), págs. 1199-1207.

[26] Bernhard Holdmann, Jean-Pierre Tillich y Peter S. L. M. Barreto. *Classic McEliece: conservative code-based cryptography: cryptosystem specification*. Submitted to the National Institute of Standards and Technology (NIST). Available online: https://csrc.nist.gov/CSRC/media/projects/post-quantum-cryptography/documents/call-for-proposals-dec-2017.pdf. 2017.

[27] W Cary Huffman y Vera Pless. *Fundamentals of error-correcting codes*. Cambridge university press, 2010.

[28] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.

[29] Kazukuni Kobara e Hideki Imai. «Semantically secure McEliece public-key cryptosystem». En: *IEICE transactions on fundamentals of electronics, communications and computer sciences* 85.1 (2002), págs. 74-83.

[30] J. B. Lee y E. F. Brickell. «An Observation on the Security of McEliece's Public-Key Cryptosystem». En: *Advances in Cryptology — EUROCRYPT '88*. Vol. 330. Lecture Notes in Computer Science. Springer, 1988, págs. 275-280. DOI: 10.1007/3-540-45961-8_24.

[31] Florence Jessie MacWilliams y Neil James Alexander Sloane. *The theory of error-correcting codes*. Vol. 16. Elsevier, 1977.

[32] Telex Magloire et al. «Two bit-flipping decoding algorithms for low-density parity-check codes». En: *IEEE transactions on communications* 57.3 (2009), págs. 591-596.

[33] Robert J McEliece. «A public-key cryptosystem based on algebraic». En: *Coding Thv* 4244 (1978), págs. 114-116.

[34] Carlos Aguilar Melchor et al. «Hamming quasi-cyclic (HQC)». En: *NIST PQC Round* 2.4 (2018), pág. 13.

[35] Rafael Misoczki et al. «MDPC-McEliece: New McEliece variants from moderate density parity-check codes». En: *2013 IEEE international symposium on information theory*. IEEE. 2013, págs. 2069-2073.

[36] Harald Niederreiter. «Knapsack-type cryptosystems and algebraic coding theory». En: *Prob. Contr. Inform. Theory* 15.2 (1986), págs. 157-166.

[37] Mohammad Reza Nosouhi et al. «Bit Flipping Key Encapsulation for the Post-Quantum Era». En: *IEEE Access* (2023).

[38] Håkon Gjeraker Østerhus. «Code-based cryptography: A superficial introduction». Tesis de mtría. The University of Bergen, 2018.

[39] Raphael Overbeck y Nicolas Sendrier. «Code-based cryptography». En: *Post-quantum cryptography*. Springer, 2009, págs. 95-145.

[40] Nicholas Patterson. «The algebraic decoding of Goppa codes». En: *IEEE Transactions on Information Theory* 21.2 (1975), págs. 203-207.

[41] David Pointcheval y Jacques Stern. «Security Proofs for Signature Schemes». En: *Advances in Cryptology — EUROCRYPT '96*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, págs. 387-398. DOI: 10.1007/3-540-68339-9_33.

[42] Marek Repka y Pavol Zajac. «Overview of the McEliece cryptosystem and its security». En: *Tatra Mountains Mathematical Publications* 60.1 (2014), págs. 57-83.

[43] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

[44] Nicolas Sendrier. «Code-Based Cryptography: State of the Art and Perspectives». En: *IEEE Security & Privacy* 15.4 (2017), págs. 44-50. ISSN: 1540-7993. DOI: 10.1109/MSP.2017.3151345. URL: http://ieeexplore.ieee.org/document/8012331/.

[45] Nicolas Sendrier. «Decoding one out of many». En: *International Workshop on Post-Quantum Cryptography*. Springer. 2011, págs. 51-67.

[46] Vladimir Michilovich Sidelnikov y Sergey O Shestakov. «On insecurity of cryptosystems based on generalized Reed-Solomon codes». En: (1992).

[47] Harshdeep Singh. *Code based Cryptography: Classic McEliece*. 29 de mayo de 2020. arXiv: 1907.12754[cs]. URL: http://arxiv.org/abs/1907.12754.

[48] National Institute of Standards y Technology. *PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*. https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4. Accessed: 2024-06-11. 2022.

[49] National Institute of Standards y Technology. *Round 3 Submissions*. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Accessed: 2024-06-06. 2017.

[50] National Institute of Standards y Technology. *Round 4 Submissions*. https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions. Accessed: 2024-06-06. 2017.

[51] National Institute of Standards y Technology. *Selected Algorithms 2022*. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Accessed: 2024-06-11. 2022.

[52] Jacques Stern. «A method for finding codewords of small weight». En: *Coding Theory and Applications: 3rd International Colloquium Toulon, France, November 2–4, 1988 Proceedings 3*. Springer. 1989, págs. 106-113.

[53] Jacques Stern. «A new identification scheme based on syndrome decoding». En: *Annual International Cryptology Conference*. Springer. 1993, págs. 13-21.

[54] Jacobus Hendricus Van Lint. *Introduction to coding theory*. Vol. 86. Springer Science & Business Media, 1998.

[55] Violetta Weger, Niklas Gassner y Joachim Rosenthal. «A survey on code-based cryptography». En: *arXiv preprint arXiv:2201.07119* (2022).

[56] Christian Wieschebrink. «Two NP-complete problems in coding theory with an application in code based cryptography». En: *2006 IEEE International Symposium on Information Theory*. IEEE. 2006, págs. 1733-1737.