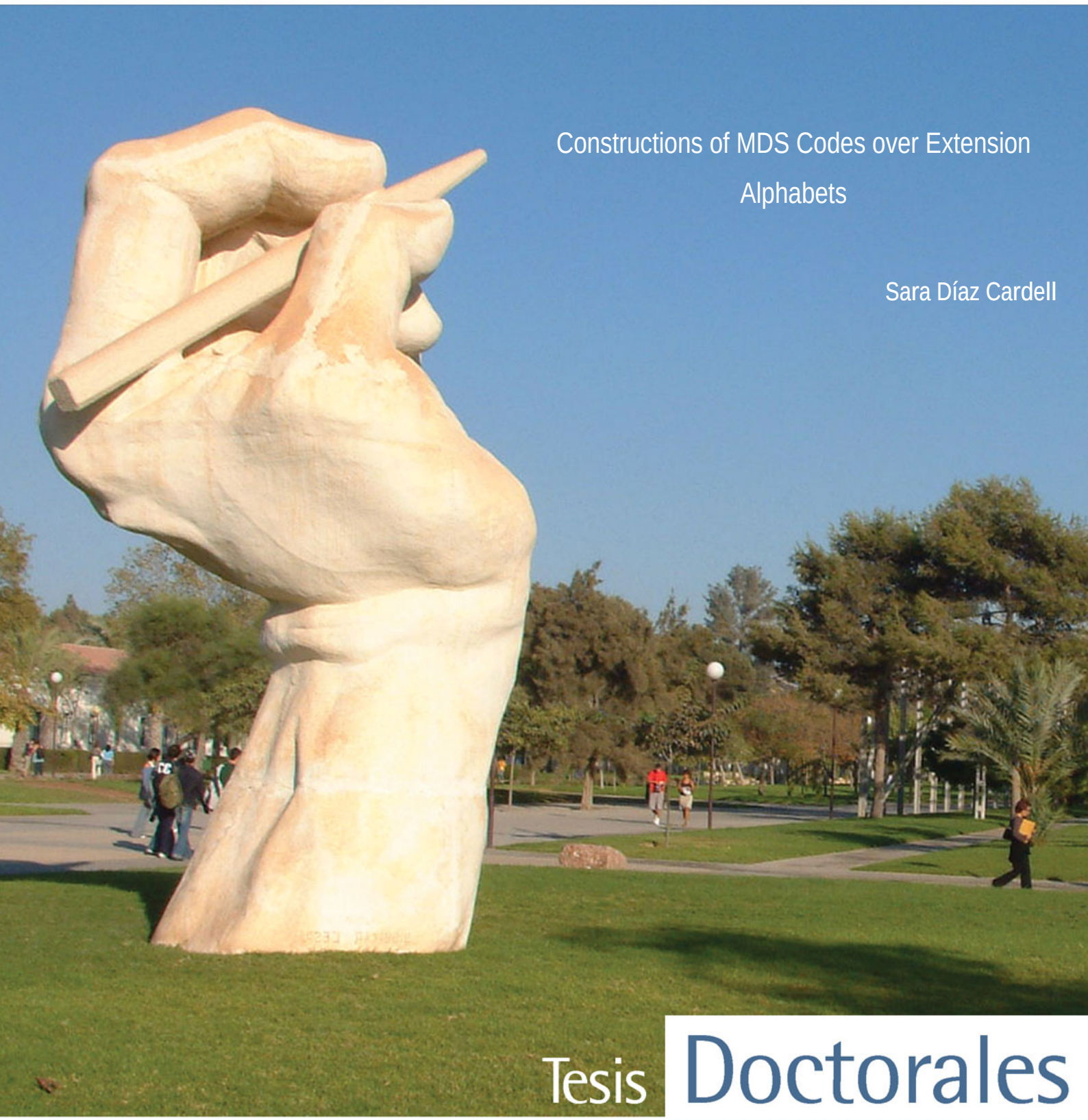
Universitat d'Alacant
Universidad de Alicante

Constructions of MDS Codes over Extension Alphabets

Sara Díaz Cardell



Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universidad de Alicante

Departamento de Ciencia de la Computación e Inteligencia Artificial

Constructions of MDS Codes over Extension Alphabets

TESIS DOCTORAL

Presentada por:
Sara Díaz Cardell

Dirigida por:
Joan Josep Climent Coloma

Universidad de Alicante

Departamento de Ciencia de la Computación e Inteligencia Artificial

Constructions of MDS Codes over Extension Alphabets

Memoria presentada para optar al grado de doctor por Sara Díaz Cardell.

Alicante, junio de 2012.

D. Joan Josep Climent Coloma, Catedrático de Universidad del Departamento de Estadística e Investigación Operativa de la Universidad de Alicante

CERTIFICA:

Que la presente memoria *Constructions of MDS Codes over Extension Alphabets*, ha sido realizada bajo su dirección, en el Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante por la licenciada Dña. Sara Díaz Cardell, y constituye su tesis para optar al grado de doctor.

Para que conste, en cumplimiento de la legislación vigente, autoriza la presentación de la referida tesis doctoral ante la comisión de Doctorado de la Universidad de Alicante, firmando el presente certificado.

Alicante, 15 de junio de 2012

Joan Josep Climent Coloma

"Cuando una persona desea realmente algo,
todo el universo conspira para que pueda realizarlo"
Paulo Coelho, El Alquimista.

A todas las personas que confiaron en mí.

Durante estos últimos cinco años he conocido a muchas personas y cada una de ellas ha aportado su granito de arena para hacer posible este trabajo. Si olvido a alguna persona, espero que me perdone.

En primer lugar, quiero agradecer a mi familia todo su apoyo y comprensión desde el momento en el que decidí estudiar Matemáticas. A pesar de que otras personas me insinuaran que era una carrera sin futuro, en casa me sentí arropada y apoyada siempre. Gracias.

Gracias a mi querido Manu por saber comprender mi situación en todo momento, por aconsejarme con el corazón y por seguir a mi lado a pesar de la distancia y los problemas.

Gracias a mis amigos por estar siempre ahí para escucharme y distraerme cuando estaba desilusionada, aunque espero que leyendo esta tesis se den cuenta de que un doctorado no es tan fácil como parece.

Quiero dar las gracias a Don Federico, mi profesor de matemáticas del colegio. Me hizo ver las matemáticas de otro modo, y desde ese momento supe que quería ser matemática. Allá donde esté, espero que me esté viendo y esté orgulloso.

Gracias a mis compañeros Virtu, Rafa y Vero. Virtu, gracias por los consejos que me diste cuando estábamos en Zürich y gracias por las interminables horas de burocracia en mi nombre. Gracias a Rafa por las charlas en nuestro antiguo despacho y por estar siempre dispuesto a ayudar. Y Vero... La mitad de esta tesis es tuya. Esta tesis no habría sido posible sin ti, sin tu apoyo y ayuda no habría sabido cómo seguir adelante en muchos momentos. No olvidaré nunca aquel verano que estuvimos trabajando y probando algoritmos juntas, los congresos, las fiestas, las horas de burocracia que has hecho en mi nombre, etc. Hace mucho tiempo que dejaste de ser una compañera de trabajo para convertirte en un amiga.

Por otro lado, gracias a Alina, Alberto y Oliver, por hacer mi vida más fácil cuando estuve en Zürich y en Dublín. En especial, gracias a Oliver por su incondicional amistad y gracias a Alberto por seguir ahí después de tantos años, apoyándome y escuchándome en mis malos momentos. Agradezco a ambos que me ayudaran con las correcciones de esta tesis.

Gracias al profesor Marcus Greferath de la University College Dublin y al profesor Joachim Rosenthal de la Universität Zürich por brindarme la oportunidad de trabajar con ellos durante varios meses. Durante ambas estancias no sólo aprendí de ellos, también crecí como persona. Fueron experiencias que me cambiaron y las cuales agradezco de corazón.

Por último, pero más importate, gracias al Doctor Joan Josep Climent. Josep, esta tesis es toda tuya, desde la primera palabra hasta la última. Confiaste en mí cuando nadie, ni siquiera yo, lo hacía. Me mostraste el camino a seguir cuando me encontraba completamente perdida y me diste ánimos en los momentos en los que pensé que no podía continuar. Eres el mejor director de tesis que cualquier doctorando podría desear. Siempre has estado dispuesto a ayudar y resolver dudas, sin hacerme sentir estúpida cuando mis preguntas eran obvias. Como le he dicho a Vero, hace tiempo que te convertiste en algo más que mi director de tesis, eres para nosotras un amigo y, a pesar de que puede que no volvamos a trabajar juntos, recordaré con mucho cariño las infinitas horas trabajando en tu despacho, las encarnizadas correcciones, los almuerzos con Vero contándonos nuestras vidas, los congresos, etc. Recuerda siempre que, tanto en la vida como en la investigación, cuando tienes la primera idea, te emocionas y luego fracasas... Cuando tienes la segunda, te emocionas y luego fracasas... Así hasta que por fin la enésima idea funciona. Sin embargo, debes saber que sin ti yo no habría llegado más alla de la primera idea. Gracias, gracias, gracias.

Contents

Prólogo

El estudio de la teoría de códigos comenzó en los años 40 del siglo pasado con los trabajos de Golay, Hamming y Shannon. A pesar de que en sus orígenes era un problema de ingeniería, se ha desarrollado cada vez más usando sofisticadas técnicas matemáticas. El principal problema de la teoría de códigos se puede describir de modo muy simple. Supongamos que una secuencia de bits se transmite a través de un canal de comunicación. Cuando hablamos de un canal nos referimos, por ejemplo, a la línea telefónica, a Internet o a un simple CD. De vez en cuando, interferencias o perturbaciones aparecen en el proceso de intercambio de datos, y algunos bits se transmiten erróneamente, es decir, recibimos un 1 en vez de un 0, o viceversa. En este punto, la cuestión es si somos capaces de identificar los errores que han ocurrido y si podemos recuperar el mensaje original.

Uno de los modelos más usuales de canal de comunicación es el canal simétrico binario, que podemos observar de forma esquemática en la figura 1. Este modelo describe una situación en la que un error ocurre a través del canal, es decir, un 1 ha sido cambiado por 0 o viceversa, con una probabilidad $p < 1/2$. De este modo, sin importar la información que entra, un error de canal, o error de bit, ocurre con

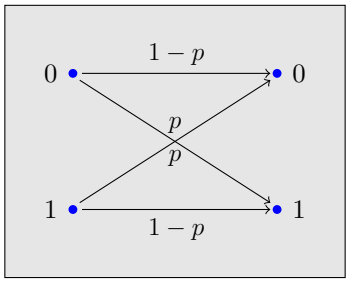


Figura 1: *Ejemplo de canal simétrico binario.*

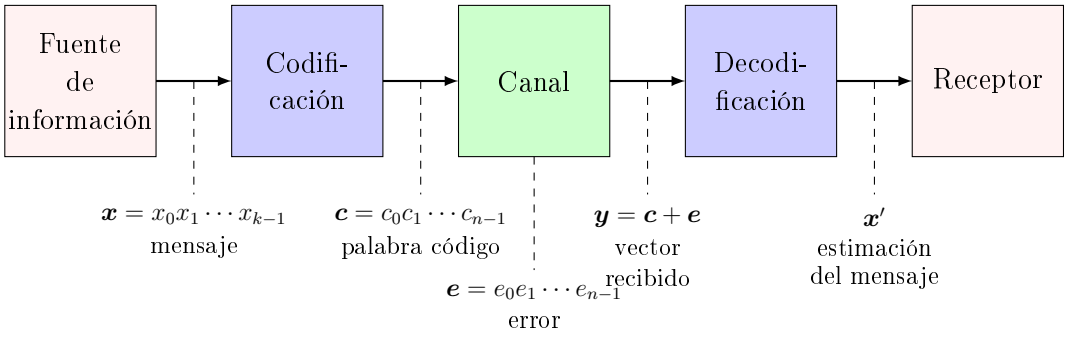


Figura 2: *Componentes del proceso de comunicación.*

probabilidad p.

Ahora imaginemos que diseñamos un esquema o algoritmo para detectar y, si es posible, corregir errores. Una posibilidad sería no hacer nada cuando recibimos un bit a la salida del canal, simplemente decidimos que es correcto. En este caso, la probabilidad de cometer un error en nuestra decisión es p.

Por otro lado, podemos ser un poco más cautos y avisar al emisor de que envíe cada bit tres veces seguidas. El receptor decidirá entonces que el bit correcto es el que aparece más veces a la salida del canal. Por ejemplo, si obtenemos como salida 001, supondremos que el bit original es 0.

A partir de esta idea, podemos pedirle al emisor que envíe cada bit un número m impar de veces seguidas. Al ser impar, siempre tendremos una mayoría de 0 o de 1. En este caso decidiremos que el bit correcto es el que aparece el mayor número de veces.

Codificar es básicamente añadir información extra o redundante, que nos ayude a detectar y corregir posibles errores en la transmisión. Sin embargo, a cambio de esto, debemos pagar un alto precio en términos de eficiencia. Por ejemplo, lleva cierta cantidad de tiempo transmitir la información a través de un canal, por lo tanto, cuanta más redundancia añadamos, más se tardará en transmitir la información a través del canal y más tardaremos en decodificar. Como consecuencia, hay que llegar a un equilibrio entre la redundancia que añadimos a la información y la capacidad del canal.

En la figura 2 podemos observar las componentes que aparecen en el proceso de comunicación. El mensaje $\boldsymbol{x} = x_0x_1\cdots x_{k-1}$ se genera en la fuente de información.

Los bits que componen el mensaje son codificados y convertidos en una palabra código $\boldsymbol{c} = c_0c_1 \cdots c_{n-1}$. Esta transformación se lleva a cabo a través de un esquema de codificación o, simplemente, código (conjunto de palabras código) y una aplicación que asigna a cada mensaje una palabra código distinta. Entonces la palabra código se transmite a través del canal, donde puede sufrir algunas modificaciones debido al error $\boldsymbol{e} = e_0e_1 \cdots e_{n-1}$. Lo que sale del canal es decodificado, es decir, se intenta detectar y corregir el máximo número posible de errores para encontrar la información original proporcionada por la fuente. Finalmente, el receptor recibe cierto mensaje de información $\boldsymbol{x}'$, que se estima que es correcto.

Los códigos fueron creados, básicamente, para corregir los errores que se producen en la información transmitida a través de canales de comunicación con ruido. Los códigos correctores de errores pueden ser divididos en *códigos de bloque* y *códigos convolucionales*.

En un código de bloque los datos se presentan en bloques de longitud fija y se añade a cada bloque un cierto número de símbolos llamados símbolos de redundancia (bits de redundancia en el caso binario). Algunos ejemplos de códigos de bloque son los códigos Reed-Solomon, Hamming, Hadamard, Golay o Reed-Muller [34, 45]. Estos ejemplos pertenecen también a la clase de códigos lineales, por lo tanto decimos que son códigos de bloque lineales. Los códigos de bloque van a tener una gran importancia en esta memoria.

Los códigos convolucionales consideran la información como una secuencia completa. Sin embargo, a diferencia de los códigos bloque, las palabras de un código convolucional se generan no sólo a partir de los dígitos de información actuales sino también con la información anterior en el tiempo. Es decir, un codificador convolucional es un sistema con memoria, un bloque de información no se codifica de forma independiente, los bloques de información anteriores afectan a bloques que se codifican posteriormente y, por lo tanto, existe solapamiento. Para más información se pueden consultar las referencias [32, 36, 46], ya que en este trabajo no vamos a abordar este tipo de códigos.

Esta memoria está estructurada de la siguiente forma. En el capítulo 1 introducimos los conceptos básicos para la comprensión del resto de capítulos. En la sección 1.2, presentamos algunos preliminares sobre códigos de bloque y códigos lineales. Un *código de bloque* es la imagen de una aplicación inyectiva $\mathcal{C} : \mathcal{A}^k \to \mathcal{A}^n$,

con $k, n \in \mathbb{N}$, donde $\mathcal{A}$ denota el alfabeto. Dados k símbolos de información, les asignamos una palabra código de longitud n (con $n > k$). Por otro lado, dado el cuerpo de Galois de q elementos $\mathbb{F}_q$, un *código lineal* es un subespacio vectorial de dimensión k de $\mathbb{F}_q^n$. Los enteros n y k son la longitud y la dimensión del código, respectivamente. Asociado a cada código lineal tenemos una matriz G llamada *matriz generadora*, cuyas filas constituyen una base del código. Existe otra matriz H asociada al mismo código llamada *matriz de control de paridad*, la cual cumple la condición $HG^T = O$.

Además, recordamos algunas propiedades interesantes que poseen los códigos lineales. En primer lugar hablamos de los códigos de *máxima distancia de separación*. Estos códigos son de vital importancia dado que corrigen el mayor número de errores posible. También recordamos la *ciclicidad*. Los códigos cíclicos son muy útiles por diversas razones. Estos códigos han sido ampliamente estudiados, puesto que su codificación es muy sencilla e incluye a la familia de los códigos BCH [34]. Por último, recordamos el concepto de código LDPC (debido a las siglas en inglés de *low density parity-check code*) [21, 26, 28, 43]. Este tipo de códigos fue introducido por Gallager [21] en los años 60 del siglo pasado. Sin embargo, hasta los años 90, no tuvieron éxito entre los investigadores. Un código LDPC es un código lineal binario cuya matriz de control de paridad es dispersa, es decir, posee un gran número de 0 y pocos 1. Por lo tanto, estos códigos se caracterizan por tener una baja densidad. Existen dos tipos de códigos LDPC: los regulares y los irregulares. Un código LDPC es regular con parámetros (i, j) si su matriz de control de paridad cumple las siguientes propiedades:

- Cada fila tiene peso constante i.
- Cada columna tiene peso constante j.
- Dos filas (respectivamente columnas) no pueden tener más de una componente en común que sea un 1.
- Tanto i como j son pequeños en comparación con el tamaño de la matriz.

Por otro lado, un código LDPC es irregular cuando los pesos de diferentes filas y diferentes columnas no son el mismo.

En la sección 1.3 introducimos un tipo de códigos de bloque llamados $\mathbb{F}_q$-*lineales*. Estos códigos son el tema principal de esta tesis. Par $b \in \mathbb{N}$, un código $\mathbb{F}_q$-lineal sobre $\mathbb{F}_q^b$, denotado por $\mathcal{C}_{\mathbb{F}_q^b}$, es un código lineal sobre $\mathbb{F}_q$, cuyas palabras código son

consideradas sobre $\mathbb{F}_q^b$, es decir, el alfabeto en este caso son vectores de longitud b. Si tenemos un código lineal $\mathcal{C}_{\mathbb{F}_q}$ de longitud nb y dimensión kb sobre $\mathbb{F}_q$, éste código puede ser a su vez considerado como un código $\mathbb{F}_q$-lineal $\mathcal{C}_{\mathbb{F}_q^b}$ de longitud n y dimensión normalizada k sobre $\mathbb{F}_q^b$. Para este tipo de códigos introducimos la generalización de conceptos como MDS o ciclicidad dados para códigos lineales. Básicamente la idea es la misma, pero tomando como alfabeto los vectores de $\mathbb{F}_q^b$ en vez de los elementos de $\mathbb{F}_q$.

Por otro lado, hablamos de la relación entre los códigos $\mathbb{F}_q$-lineales considerados en esta memoria o en otros trabajos como [7, 33] y los llamados *códigos array* [1, 2, 4, 6]. Los códigos array son un tipo de código corrector de errores, que han sido ampliamente estudiados por varios autores en el contexto de almacenamiento de datos y sistemas de comunicación [1, 3, 6, 60, 61]. Los códigos array más importantes a considerar son los binarios de dimensión dos. Estos códigos se diferencian de otros códigos, como los Reed-Solomon en el hecho de que sus algoritmos de codificación y decodificación usan sólo operaciones módulo 2, que pueden ser implentadas fácilmente en hardware y en software. En general, los códigos array tienden a requerir menos complejidad que los códigos basados en cuerpos finitos.

Los códigos array que se consideran normalmente son arrays (matrices) de tamaño $b \times n$ en $\mathbb{F}_2$. Asumimos que k columnas llevan la información, y las restantes $n-k$ columnas son columnas de paridad. De este modo podemos ver el código array como un código de longitud n y dimensión k sobre $\mathbb{F}_2^b$. La distancia mínima d es el mínimo número de columas no nulas para un array (matriz) en el código.

Por ejemplo, supongamos que las matrices

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

forman un código array. En este caso $b = 2$ y $n = 3$. Supongamos que las primeras $k = 2$ columnas llevan la información. Es posible ver estas matrices como las palabras código

$$[\ 00 \quad 00 \quad 00\ ], \quad [\ 10 \quad 10 \quad 10\ ], \quad [\ 01 \quad 01 \quad 01\ ], \quad [\ 11 \quad 11 \quad 11\ ],$$

que forman un código de longitud $n = 3$ sobre $\mathbb{F}_2^2$. De hecho, las palabras anteriores forman un código $\mathbb{F}_2$-lineal sobre $\mathbb{F}_2^2$.

En la sección 1.4 recordamos algunos conceptos relacionados con la criptografía. Los sistemas de cifrado de clave privada o simétricos son una clase importante dentro de los diferentes tipos de sistemas de cifrado [9, 17, 39]. Generalmente se dice que son mucho más rápidos que los de clave pública, como por ejemplo el RSA [8, 39, 44]. Se caracterizan por el hecho de que la misma clave es usada para cifrar y descifrar, de este modo, emisor y receptor deben compartir la clave secreta antes de comenzar a transmitir información. El intercambio de la clave entre ellos representa un papel crucial en la comunicación. Si un intruso se hiciera con la clave podría descrifrar toda la información transmitida entre las dos partes. Supongamos que el mensaje es muy largo; en el caso de que la primera componente o letra del mensaje dependiera de la última el proceso de transmisión sería bastante lento. Tampoco durante el descifrado se desea tener que esperar a que la secuencia entera de texto cifrado sea transmitida a través del canal antes de empezar a descifrar. La solución es, por supuesto, dividir el mensaje en pequeños bloques, donde procesar cada bloque depende de ese mismo bloque y posiblemente de algunos bloques previos, ya procesados. Para conseguir esto, hay dos clases de sistemas de cifrado, *cifrador en bloque* y *cifrador en flujo*. Para hacer la discusión más precisa, asumimos que los mensajes son secuencias binarias. Esto no implica ninguna restricción en los tipos de mensajes a cifrar, ya que todo tipo de infomación puede ser representada como secuencia binaria.

En los cifradores en flujo, el tamaño de cada bloque es normalmente el tamaño de una de las letras del alfabeto con el que se está trabajando. Como estamos trabajando sobre el alfabeto binario, el cifrado se produce bit a bit. Al contrario que en los cifradores en bloque, donde el tiempo es el mismo para cada bloque, el tiempo en el cifrado de un cifrador en flujo varía, es decir, el cifrado varía de bit a bit. Además cada bit es cifrado de forma separada e independiente del resto de bits. Por esto, si hay patrones en el mensaje no se reconocen en el texto cifrado. Otra ventaja es que el error de propagación es realmente limitado.

Si conseguimos, con una clave secreta de longitud l, generar una secuencia $\boldsymbol{r} = (r_0, r_1, \ldots, r_{N-1})$ que parezca aleatoria de longitud N, llamada *secuencia clave*, podríamos cifrar un mensaje de longitud N. Para cifrar un mensaje m lo sumamos con la secuencia clave y obtenemos así el mensaje cifrado c. Para descifrar procedemos de modo similar, sumamos el texto cifrado c y la secuencia clave y de este modo volvemos a obtener m. Los cifradores en flujo que utilizan este método son bastante populares y se llaman *aditivos*. Un ejemplo muy famoso de un cifrador en

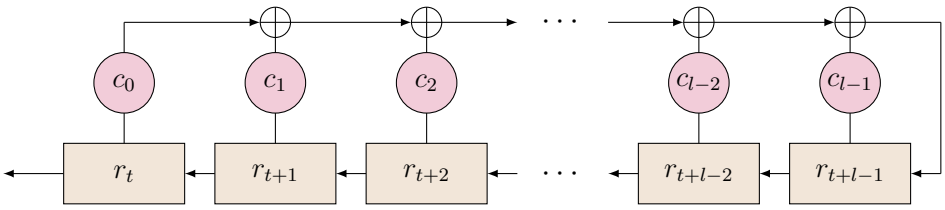


Figura 3: *Estructura de un LFSR de longitud l.*

flujo aditivo es el llamado *one-time pad* [39, 51].

El primer problema a resolver es generar un secuencia que parezca aleatoria $\boldsymbol{r} = (r_0, r_1, r_2, \ldots)$ usando una clave pequeña. Los algoritmos que generan secuencias clave usando una clave secreta se llaman *generadores de secuencias clave*. Estamos interesados en los que están basados en registros de desplazamiento con retroalimentación lineal, más conocidos por sus siglas en inglés LFSR (*linear feedback shift registers*) [18, 19, 58]. Los LFSR generan secuencias clave de forma lineal, lo que hace su computación muy eficiente. Por otro lado, su linealidad los hace fácilmente predecibles, por esta razón no se utilizan directamente como generadores de secuencias clave, sino que se utilizan como base para otros más avanzados.

Un LFSR de longitud l consiste en l registros, llamados con la palabra inglesa *taps*. Cada tap tiene un coeficiente asociado llamado *coeficiente de retroalimentación* $c_i \in \mathbb{F}_2$, para $i = 0, 1, 2, \ldots, l-1$ que define si el bit en ese tap se usa en los cálculos del la suma o no. En la figura 3 podemos observar la estructura de un LFSR.

Sea $\boldsymbol{r} = (r_0, r_1, r_2, \ldots)$ la secuencia obtenida a partir de un LFSR. Entonces $\boldsymbol{r}$ satisface la relación recurrente

$$c_0 r_t + c_1 r_{t+1} + c_2 r_{t+2} + \cdots + c_{l-1} r_{t+l-1} + r_{t+l} = 0, \quad t \geq 0.$$

El polinomio $f(\mathsf{x}) = c_0 + c_1\mathsf{x} + c_2\mathsf{x}^2 + \cdots + c_{l-1}\mathsf{x}^{l-1} + \mathsf{x}^l$ se llama *polinomio de retroalimentación* del LFSR. Una secuencia $\boldsymbol{r}$ generada por el LFSR está completamente determinada por los l valores iniciales $\boldsymbol{R}_0 = (r_0, r_2, \ldots, r_{l-1})$, llamado *estado inicial*. Así el número de las diferentes secuencias que un LFSR puede generar es 2^l. Los valores de los taps en el instante t, $\boldsymbol{R}_t = (r_t, r_{t+1}, \ldots, r_{t+l-1})$ se llama *estado* t del LFSR. Lo que se pretende es esconder mensajes lo más largo posibles. El periodo de un LFSR cuyo polinomio de retroalimentación es un polinomio primitivo $f(\mathsf{x}) \in \mathbb{F}_2[\mathsf{x}]$ de grado l, es $2^l - 1$, siempre que el estado inicial no sea el cero. Este periodo es el máximo periodo posible [31].

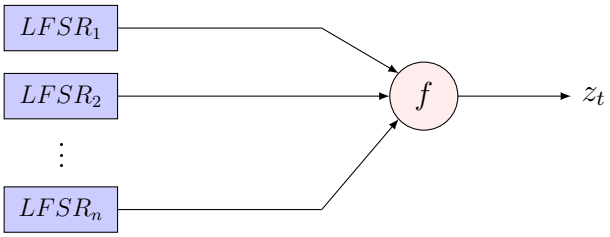


Figura 4: *Modelo que combina varios LFSR a través de un función booleana.*

Por otro lado, *la complejidad lineal* de una secuencia dada es la longitud del LFSR más corto que la genera. La complejidad lineal puede determinarse con el algoritmo de Berlekamp-Massey [35], que calcula el polinomio de retroalimentación del LFSR con al menos el doble de bits de la complejidad lineal. Por esto, la salida de un LSFR no puede tomarse como secuencia de cifrado. Las secuencias que se obtienen tienen buenas propiedades, algunas son deseables para una secuencia de cifrado, pero el problema radica en la linealidad. Por ello, se intenta aumentar la complejidad lineal antes de usar la secuencia. Una idea bastante clásica, es combinar la salida de varios LFSR a través de una función booleana no lineal f. En la figura 4 podemos observar la idea de este modelo que comentamos antes para romper la linealidad de un LFSR.

Recordemos que una función booleana es una applicación $f : \mathbb{F}_2^n \to \mathbb{F}_2$, con $n \in \mathbb{N}$. Las propiedades de la secuencia de salida dependen en gran medida de la elección de la función f. Se desearía que esta función tuviera propiedades tales como un alto grado algebraico, una alta no linealidad y que fuera inmune a la correlación (véase [22, 42, 50, 58]).

Cuando elegimos una función booleana que no satisface estas propiedades, se puede obtener el estado inicial del modelo, a través de un ataque de correlación, como veremos en la sección 1.5.

En el capítulo 2 introducimos una construcción de códigos $\mathbb{F}_2$-lineales que además son MDS, cíclicos y con baja densidad sobre $\mathbb{F}_2^b$. Para este propósito introducimos en la sección 2.1 el concepto de *array de índices*. Se trata de una tabla en la cual almacenamos la posición de los 1 de una matriz, principalmente la matriz de control de paridad. Cada celda de la tabla contiene la posición de los 1 de una columna diferente. También necesitamos introducir el concepto de logaritmo de Zech. Dado

un elemento primitivo $\alpha \in \mathbb{F}_p$, con p primo, denotamos el logaritmo de Zech de un elemento $x \in \mathbb{Z}_{p-1}$ como $\mathcal{Z}_\alpha(x)$. El logaritmo debe cumplir que $\alpha^{\mathcal{Z}_\alpha(x)} = 1 + \alpha^x$ y suponemos $\alpha^\infty = 0$.

En la sección 2.3 explicamos nuestra construcción. Elegimos un número primo p tal que $p - 1 = rb$. Construimos un array de índices de tamaño $b \times (p-1)$ que representa una matriz de control de paridad de un código $\mathbb{F}_2$-lineal con ciertas propiedades en $\mathbb{F}_2^b$. Para ello primero construimos los conjuntos

$$E_i = \{x \in \mathbb{Z}_{p-1} \mid i = x \text{ mód } b\}, \quad \text{para} \quad i = 0, 1, \ldots, b-1,$$

que forman una partición de $\mathbb{Z}_{p-1}$. A continuación, construimos los conjuntos $D_i = \{\mathcal{Z}_\alpha(x) \mid x \in E_i\}$ para $i = 0, 1, \ldots, b-1$ con $i \neq \frac{p-1}{2}$ mód b. Finalmente construimos un array de índices de dimensiones $b \times (p-1)$. En la fila 0 del array colocamos los números $0, 1, \ldots, p-2$, con lo que el array tendrá $p-1$ columnas. En la columna 0 colocamos por orden los conjuntos D_i. Supongamos que empezamos a contar filas y columnas en 0, entonces en el resto de columnas, por ejemplo en la columna t colocamos los conjuntos $D_i + t$ en dicho orden. Por $D_i + t$ entendemos el conjunto que se obtiene al sumar t módulo $p-1$ a todos los elementos del conjunto D_i. El array resultante representa la posición de los 1 para una matriz de control de paridad de un código $\mathbb{F}_2$-lineal sobre $\mathbb{F}_2^b$. La longitud de este código será $p-1$ y su dimensión normalizada será $p-1-r$.

En la sección 2.4 estudiamos las propiedades de este tipo de códigos. Dada la forma cíclica del array, podemos afirmar que el código es cíclico. La propiedad MDS no es tan fácil de probar. Para $r = 2$, todo parece indicar que los códigos que construimos son MDS. Sin embargo, no hemos llegado a probar esta parte. Esto es consecuencia de la forma en que están construidas las columnas del array, aunque dado el carácter aleatorio de los logaritmos discretos como el logaritmo de Zech, no hemos conseguido todavía un resultado satisfactorio. Por otro lado, los códigos tienen una baja densidad, es decir son LDPC, pero irregulares, ya que todas las columnas no tienen la misma cantidad de 1. La primera columna de cada bloque posee un solo 1, mientras que el número de 1 en el resto de columnas es r. Es cierto que con la elección de r estamos eligiendo el número de 1 que va a tener la matriz, controlamos este dato, pero no podemos conseguir que sea un código regular. Básicamente la matriz de control de paridad tiene $p-1$ columnas con un 1 y, para las restantes $(p-1)(b-1)$ columnas el número de 1 es r, es decir, en total el número

de 1 de la matriz será $(p-1)(1+(b-1)r) = (p-1)(p-r)$. Teniendo en cuenta que la matriz posee $(p-1)^2 b$ elementos, tenemos que la media de 1 en la matriz será

$$\frac{p^2 - p(r+1) + r}{(p-1)^2 b} = \frac{rp^2 - pr(r+1) + r^2}{(p-1)^3}.$$

Considerando el caso $r = 2$, que es cuando el código es MDS, tenemos que la media es

$$\frac{2p^2 - 6p + 4}{(p-1)^3} = \frac{2(p-2)}{(p-1)^2}.$$

Cuando p se hace grande, esta media tiende a 0, es decir, cuanto mayor sea la longitud del código que estamos construyendo, más baja será la densidad de la matriz de control de paridad.

Además de la construcción de este tipo de códigos, en este capítulo introducimos varios algoritmos más. En la sección 2.2 introducimos una nueva forma de aplicar el método de Gauss-Jordan para matrices binarias. Si en vez de tener la matriz, lo que tenemos es el array de índices podemos aplicar el método de Gauss-Jordan para comprobar si nuestra matriz es singular o no. Si nuestro array posee en sus celdas índices del 0 a $p-2$, empezamos eligiendo una celda que posea el 0. Con esta celda eliminamos los 0 de otras celdas, arrastrando los elementos que estuvieran con el 0 en la celda elegida y eliminando alguno de esos elementos si ya estuviera. Por ejemplo, supongamos que tenemos una celda que posee los índices $\{0, 3, 5\}$ y con ella queremos eliminar el 0 del resto de celdas. Supongamos que otra celda contiene los índices $\{0, 4, 5\}$. Entonces la celda resultante de eliminar el 0 sería $\{3, 4\}$. Hemos eliminado tanto el 0 como el 5, porque aparecían en las dos celdas y nos hemos quedado con los elementos no comunes. Llevaríamos a cabo el mismo proceso para $1, 2, \ldots, p-2$. Al final del proceso obtendríamos un array en el que cada celda contendría un único índice, todos ellos distintos y comprendidos entre 0 y $p-2$, en el caso de que este array represente a una matriz no singular, u obtendríamos un array con alguna de las celdas vacías, en el caso de una matriz singular. En la sección 2.5 introducimos cómo obtener el array que representa la matriz generadora de nuestros códigos a partir del array de la matriz de control de paridad.

Por último en la sección 2.6 introducimos un algoritmo de decodificación a través del array de índices. Explicamos este algoritmo para nuestro caso binario cuando $r = 2$, pero puede ser ampliado para otro código, siempre que sepamos cuántos errores corrige y conozcamos el array de índices de su matriz de control de paridad.

Básicamente, colocamos cada bit de la palabra código en su celda correspondiente. Nos quedamos con las celdas cuyo bit correspondiente es 1. Ahora contamos el número de veces que aparece cada índice. El síndrome será el vector cuyas componentes son el número de veces que aparece cada índice $0, 1, \ldots, p-2$ módulo 2, ya que estamos en el caso binario. Ahora, si la $i$ésima y la $j$ésima componentes son, por ejemplo, 1 (empezando a contar en 0) sabemos que hay un error en una de las columas del array donde aparecen los índices i y j juntos, o solos, sin afectar a los demás. Sabiendo el número de la columna, por ejemplo la t, sabemos que el símbolo que aparece en la posición t de la palabra es erróneo.

En el capítulo 3 introducimos otra construcción de códigos $\mathbb{F}_q$-lineales, que en este caso serán MDS. Esta construcción se basa en el concepto de *matriz superregular* [47]. Una matriz superregular es una matriz tal que todas sus submatrices son no singulares. Las matrices de Cauchy o las de Vandermonde son ejemplos de este tipo de matrices [29, 30]. Dado un código lineal $\mathcal{C}_{\mathbb{F}_q}$ sobre $\mathbb{F}_q$, si la matriz de control de paridad (respectivamente, la generadora) está en forma sistemática, es decir, $H = [\ A \quad I\ ]$ (respectivamente $G = [\ I \quad B\ ]$), el código es MDS si y sólo si la matriz A (respectivamente, la matriz B) es superregular [34]. Nosotros nos basamos en esta idea, para intentar construir códigos $\mathbb{F}_q$-lineales MDS sobre $\mathbb{F}_q^b$. En este punto introducimos el concepto de *matriz superregular por bloques* de tamaño $b \times b$. La idea es la misma, simplemente en este caso no todas las submatrices tienen que ser no singulares, sólo las formadas por bloques principales de tamaño $b \times b$. Por ejemplo, la matriz

$$\left[\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

es superregular por bloques de tamaño 2×2, dado que las submatrices

$$\left[\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array}\right], \quad \left[\begin{array}{cc} 1 & 1 \\ 0 & 1 \end{array}\right], \quad \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right], \quad \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array}\right],$$

y la propia matriz son no singulares.

Un código $\mathbb{F}_q$-lineal $\mathcal{C}_{\mathbb{F}_q^b}$ sobre $\mathbb{F}_q^b$ cuya matriz de control de paridad está en forma sistemática $H = [\ A \quad I\ ]$ es MDS si y sólo si la matriz A es superregular por bloques

de tamaño $b \times b$ [7]. Nuestra construcción consiste en obtener matrices de control de forma sistemática $H = [\ A \quad I\ ]$ tal que la matriz A es superregular por bloques.

En las secciones 3.2 y 3.3 mostramos las construcciones propuestas. Consideramos la matriz companion C de un polinomio primitivo de grado b con coeficientes en $\mathbb{F}_q$. El conjunto $\mathbb{F}_q[C] = \{I, C, C^2, \ldots, C^{q^b-2}\}$ es isomorfo al cuerpo $\mathbb{F}_q^b$ (véase [31]). Podemos construir un isomorfismo ψ entre estos dos cuerpos de modo que $\psi(\alpha) = C$, siendo $\alpha \in \mathbb{F}_q^b$ un elemento primitivo. Dada una matriz superregular M de tamaño $t \times m$ en $\mathbb{F}_q^b$, podemos obtener una matriz de tamaño $bt \times bm$ en $\mathbb{F}_q$, superregular por bloques de tamaño b, sustituyendo cada elemento en la matriz M por su correspondiente imagen dado el isomorfismo ψ. La nueva matriz será la matriz A en $H = [\ A \quad I\ ]$. Con esto nos aseguramos que nuestro código sea MDS.

Sabemos que las matrices de Cauchy y de Vandermonde son matrices superregulares, por lo que en la sección 3.4 a partir de ellas obtenemos familias de matrices superregulares por bloques, simplemente aplicando el isomorfismo obtenido anteriormente en la sección 3.3.

En la sección 3.5 introducimos un algoritmo de decodificación para este tipo de códigos. Se basa en el algoritmo propuesto por Blaum, Bruck y Vardy en [3]. Básicamente consideramos cada bloque de una palabra código $\boldsymbol{c}$ como un polinomio con variable α y en vez de considerar la matriz de control de paridad tal cual la conocemos, consideramos $\psi^{-1}(H)$, donde ψ es el isomorfismo propuesto anteriormente. Calculamos el síndrome computando $\psi^{-1}(H)\boldsymbol{c}^T$. Como las palabras código, el síndrome estará dividido por bloques, dependiendo de cuántos bloques sean nulos y de que cumplan ciertas condiciones sabremos si los errores que buscamos están en los símbolos de paridad o de información. A veces simplemente el algoritmo no puede corregir, ya que no corrige más allá de su capacidad correctora.

El capítulo 4 se centra en los *LFSR* y en los *LFSR generalizados*. Dado el polinomio de retroalimentación $p(\mathsf{x}) = c_0 + c_1\mathsf{x} + \cdots + c_{l-1}\mathsf{x}^{l-1} + \mathsf{x}^l \in \mathbb{F}_2[\mathsf{x}]$ de un LFSR, la secuencia de salida se puede plantear como la salida del sistema autónomo

$$\begin{cases} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t A, \\ y_t &= \boldsymbol{x}_t C, \end{cases}$$

donde A es la matriz companion del polinomio de retroalimentación y C es la primera

columna de la matriz identidad. Este sistema se puede representar a su vez como

$$\boldsymbol{y} = \boldsymbol{x}_0 \left[\begin{array}{cccc} C & AC & A^2C & \cdots & A^{N-1}C \end{array} \right] = \boldsymbol{x}_0 G.$$

Para un LFSR generalizado, la idea es la misma, pero ahora, en vez de tener un polinomio de retroalimentación, tenemos una matriz polinómica de tamaño $m \times m$, $P(\mathsf{x}) = \sum_{i=0}^{\delta} P_i \mathsf{x}^i$ con $P_\delta = I_m$, cuyo determinante $\det P(\mathsf{x}) = p(\mathsf{x})$ es un polinomio primitivo. En este caso, consideramos la matriz companion por bloques

$$A = \left[\begin{array}{cccc:c} O & O & \cdots & O & P_0 \\ I_m & O & \cdots & O & P_1 \\ O & I_m & \cdots & O & P_2 \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \cdots & O & P_{\delta-2} \\ O & O & \cdots & I_m & P_{\delta-1} \end{array} \right]$$

y $C = \left[\begin{array}{c} I_m \\ O \end{array} \right]$. El sistema sería de la misma forma que en el caso LFSR. Analizamos los diferentes casos posibles para diferentes matrices polinómicas, ya que dependiendo del tamaño de la matriz $P(\mathsf{x})$ el tamaño de la secuencia de salida puede variar.

Podemos estudiar los LFSR como códigos Simplex o como códigos $\mathbb{F}_2$-lineales MDS, donde G es la matriz generadora. Al estar esta matriz en forma sistemática $G = [\ I \quad B\ ]$ y ser B una matriz formada por potencias de una matriz companion de un polinomio primitivo, el código es MDS. Entonces, una secuencia de salida de un LFSR se puede considerar como una palabra código de un código $\mathbb{F}_2$-lineal. El primer bloque de información, sería el estado inicial o clave del LFSR, y el resto de la palabra código se puede considerar como los bloques de paridad. Lo que nos interesa recuperar es el bloque de información. Planteamos un algoritmo de decodificación basado en el síndrome, considerando el síndrome como un vector por bloques. Dependiendo del bloque que no es nulo sabremos la posición del error, si está en el símbolo de información o en los de paridad.

Por último, incluimos un apéndice con conclusiones y líneas futuras, y las referencias utilizadas para la elaboración de este trabajo, las cuales hemos citado a lo largo del texto.

Este prólogo constituye un resumen del trabajo presentado en esta tesis escrito en una de las dos lenguas oficiales de la Comunidad Valenciana, de acuerdo con la normativa propia de la Universidad de Alicante. El resto del contenido está escrito en inglés para poder optar a la mención de Doctor Internacional.

Este trabajo ha sido subvencionado por las ayudas y proyectos siguientes:

- Ayuda para becas y contratos destinada a la formación de doctores concedida por la Generalitat Valenciana con referencia BFPI/2008/138.
- Proyecto I+D: Construcción de códigos convolucionales. Algoritmos secuenciales y paralelos de decodificación (MTM2005-05759). Subvencionado por el Ministerio de Educación y Ciencia.
- Proyecto I+D: Construcción de funciones bent y códigos LDPC. Aplicaciones criptográficas. (MTM2008-06674-C02- 01). Subvencionado por el Ministerio de Ciencia e Innovación.
- Proyecto I+D: Estudio, construcción e implementación de turbo códigos, códigos convolucionales y códigos LDPC. Aplicaciones criptográficas (MTM2011-24858). Subvencionado por el Ministerio de Economía y Competitividad.
- Proyecto I+D: Criptología y seguridad computacional (VIGROB-025). Subvencionado por la Univesidad de Alicante.

Una parte de los resultados obtenidos en este trabajo han sido presentados y pueden encontrarse en las actas de los siguientes congresos, conferencias o workshops:

- 5th Workshop on Coding and Systems, Dublín, Irlanda, 2009.
- XXI Congreso de Ecuaciones Diferenciales y Aplicaciones / XI Congreso de Matemática Aplicada, Ciudad Real, España, 2009.
- 10th International Conference on Computational and Mathematical Methods in Science and Engineering, Almería, España, 2010.
- 19th International Symposium on Mathematical Theory of Networks and Systems, Budapest, Hungría, 2010.
- 6th Workshop on Coding and Systems, Aveiro, Portugal, 2011.
- 11th International Conference on Computational and Mathematical Methods in Science and Engineering, Benidorm, España, 2011.
- 3rd International Castle Meeting on Coding Theory and Applications, Barcelona, España, 2011.
- Workshop on Codes and Topology, Castro Urdiales, España, 2012.

Chapter 1

Preliminaries

1.1 Introduction

In 1948 Claude Shannon, considered father of modern Information, published "A mathematical theory of communication" [52]. It signified the beginning of information theory and coding theory. Information is sent over a communication channel and this information may be corrupted due to the noise in the channel. Shannon identified a number called the capacity of the channel and proved that reliable communication is possible under the rate of the capacity [24]. Shannon said that information can be encoded before transmission so that the corrupted data can be decoded. The most usual model of information channel is illustrated in Figure 1.1. That is the binary symmetric channel.

Assume a sequence composed by 1s and 0s is transmitted through a communication channel. Sometimes some bits are flipped during the process. The main problem considered is to detect these errors and correct them. Coding theory appears as a

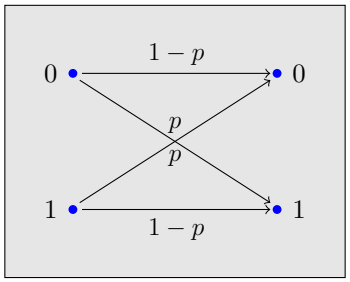


Figure 1.1: *Example of binary symmetric channel.*

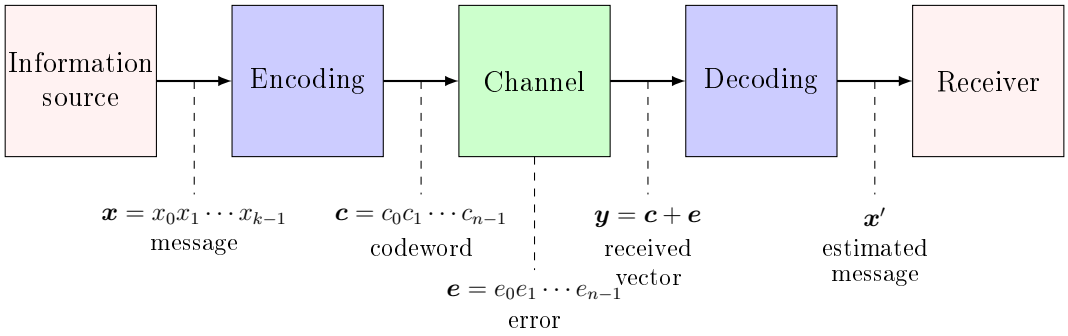


Figure 1.2: *Components of the communication process.*

solution to this problem. Error correcting codes are used to improve the reliability of communication systems. The purpose is to add redundancy to the information in order to recover the information as accurately as possible after transmitting (see Figure 1.2).

In Section 1.2 we recall the definition of block code, linear code and the properties of cyclicity, low density and maximum distance separation [24, 34]. In Section 1.3 we introduce a new type of code, the $\mathbb{F}_q$-linear code. These codes are linear over $\mathbb{F}_q$, with $\mathbb{F}_q$ the Galois field of q elements, but the alphabet to be considered is $\mathbb{F}_q^b$ (see, for instance, [7, 33]). The properties considered in Section 1.2 can be extended for this kind of codes.

In Section 1.4 we remember some concepts related with cryptography. Symmetric key ciphers, also known as secret key ciphers, are an important class of cipher systems [39, 51]. They are characterized by the fact that the same key is used for encryption and decryption and they are divided into block ciphers or stream ciphers.

In stream ciphers, the block size to be encrypted normally equals the size of one character in the alphabet. If we use the binary alphabet $\{0, 1\}$, the encryption is done bit by bit. Contrary to block ciphers, where the encryption is the same for each block, the encryption in stream ciphers is time-varying, the encryption varies from bit to bit. Besides, every bit is encrypted separately and independently from the previous or the following bits. Consequently, patterns in the plaintext are not recognizable in the ciphertext and error propagation is very limited. Many of the properties of stream ciphers make them suitable for use in telecommunication and low-level network.

They are normally much faster than block ciphers and do not cost more to implement in terms of hardware gates or memory. If we can, having a secret key k of length l, generate a random looking sequence $\boldsymbol{r} = (r_0, r_1, \ldots, r_{N-1})$ of length N, called **keystream**, we can encrypt messages of length N. Encryption is done by just adding $\boldsymbol{r}$ to the message $\boldsymbol{m}$ modulo 2,

$$c_i = m_i + r_i, \quad i = 0, 1, \ldots, N-1,$$

and decryption by again adding $\boldsymbol{r}$ to the ciphertext $\boldsymbol{c}$

$$m_i = c_i + r_i, \quad i = 0, 1, \ldots, N-1.$$

In fact, this is a widely used approach for stream ciphers, called **additive stream ciphers**. A famous example of an additive stream cipher is the well-known *one-time pad* [39, 51].

In Section 1.4 we recall the concept of linear feedback shift register (LFSR), a linear generator for pseudorandom sequences and the idea of linear complexity of a sequence. Finally, in Section 1.5 we speak about the idea of correlation attacks.

1.2 Block Codes

In this section we recall some essential ideas of coding theory. We start with the definition of block code and Hamming distance and we continue with linear codes, decoding and MDS codes.

1.2.1 Introduction

A **block code** $\mathcal{C}$ is the image of an injective map

$$\varphi : \mathcal{A}^k \longrightarrow \mathcal{A}^n, \quad \text{where } k, n \in \mathbb{N},$$

and the set $\mathcal{A}$ is called the **alphabet**. Let $\mathbb{F}_q$ be the Galois field of q elements. In this section we consider $\mathcal{A} = \mathbb{F}_q$. If $\mathbb{F}_q^n$ is the vector space of all n-tuples over the finite field $\mathbb{F}_q$, we define a code over $\mathbb{F}_q$ as follows.

Definition 1.1: An (n, M) code $\mathcal{C}$ over $\mathbb{F}_q$ is a subset of $\mathbb{F}_q^n$ of size M, that is, the number of codewords it contains.

The vectors $[\ c_0 \quad c_1 \quad \dots \quad c_{n-1}\ ] \in \mathbb{F}_q^n$ are called **words**. We sometimes write the vectors in the form $c_0c_1\cdots c_{n-1}$ and call the vectors in $\mathcal{C}$ as **codewords**.

When $q = 2$ the code is said to be a binary code. In this work, binary codes will be of great importance.

Example 1.1: We consider the code $\mathcal{C}$ over $\mathbb{F}_2$,

$$\mathcal{C} = \{0001, 0010, 1011\}.$$

This code has size $M = 3$, since it contains 3 codewords, and the length of the codewords is $n = 4$. Then, $\mathcal{C}$ is a $(4, 3)$-code over $\mathbb{F}_2$. ■

Now, we recall the concept of distance and Hamming distance.

Definition 1.2: A function $d : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{R}$ is said to be a **distance function** if it satisfies the following properties:

(a) $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$.
(b) $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$.
(c) $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$.
(d) $d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$ for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{F}_q^n$.

Definition 1.3: For $\boldsymbol{x} = [\ x_0 \quad x_1 \quad \cdots \quad x_{n-1}\ ]$ and $\boldsymbol{y} = [\ y_0 \quad y_1 \quad \cdots \quad y_{n-1}\ ]$ in $\mathbb{F}_q^n$ the **Hamming distance** $d(\boldsymbol{x}, \boldsymbol{y})$ between the words $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined by

$$d(\boldsymbol{x}, \boldsymbol{y}) = |\{i \mid x_i \neq y_i, i = 0, 1, \dots, n-1\}|.$$

Basically, the Hamming distance is the number of places where $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. It is easy to check that the Hamming distance satisfies the properties given in Definition 1.2. From now on, we consider only the Hamming distance. The **minimum distance** of a code $\mathcal{C}$ is the smallest distance between any two codewords in $\mathcal{C}$

$$d = \min\{d(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}\}.$$

The minimum distance is very important in determining the error-correcting capacity of the code. As we will see in Section 1.2.3, the higher the minimum distance is, the more errors the code can correct.

Definition 1.4: The **Hamming weight** $\omega(\boldsymbol{x})$ of $\boldsymbol{x} \in \mathbb{F}_q^n$ is the number of nonzero coordinates in $\boldsymbol{x}$.

Let $A_q(n, d)$ be the maximum number of codewords in a code over $\mathbb{F}_q$ of length n and distance at least d. The following theorem introduces the Singleton bound. We can find the proof in [24, 34], for example.

Theorem 1.1 (Singleton bound): *For $d \leq n$, we have*

$$A_q(n, d) \leq q^{n-d+1}. \tag{1.1}$$

1.2.2 Linear Codes

The next definition introduces the concept of linear code over $\mathbb{F}_q$.

Definition 1.5: Let $\mathbb{F}_q^n$ denote the n-dimensional vector space over $\mathbb{F}_q$. A k-dimensional subspace of $\mathbb{F}_q^n$ is said to be a **linear code** $\mathcal{C}_{\mathbb{F}_q}$ with parameters $[n, k]$ over $\mathbb{F}_q$. The parameter n is called the length of the code and k is the dimension.

It is worth noticing that a linear code with parameters $[n, k]$ over $\mathbb{F}_q$ contains q^k codewords.

The next theorem shows us the relationship between distance and weight. We can find the proof in [24, 34], for instance.

Theorem 1.2: *For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, $d(\boldsymbol{x}, \boldsymbol{y}) = \omega(\boldsymbol{x} - \boldsymbol{y})$. If $\mathcal{C}_{\mathbb{F}_q}$ is a linear code over $\mathbb{F}_q$, the minimum distance d of $\mathcal{C}_{\mathbb{F}_q}$ is the same as the minimum weight of the nonzero codewords of $\mathcal{C}_{\mathbb{F}_q}$.*

Then, for a linear code, the minimum distance is also called the minimum weight of the code. If d is the minimum distance of an $[n, k]$ code, we can refer to the code as an $[n, k, d]$ code.

A **generator matrix** for an $[n, k]$ linear code $\mathcal{C}_{\mathbb{F}_q}$ is any matrix G of size $k \times n$

whose rows form a basis of the code $\mathcal{C}_{\mathbb{F}_q}$, that is,

$$\mathcal{C}_{\mathbb{F}_q} = \{\boldsymbol{u}G \mid \boldsymbol{u} \in \mathbb{F}_q^k\}.$$

On the other hand, we can define the **parity-check matrix** of the code $\mathcal{C}_{\mathbb{F}_q}$ as a matrix H of size $(n-k) \times n$ such that,

$$\mathcal{C}_{\mathbb{F}_q} = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid H\boldsymbol{x}^T = \boldsymbol{0}\}.$$

The rows of H must be independent as well.

The set $\mathcal{C}_{\mathbb{F}_q}^{\perp}$ of all vectors orthogonal to every vector in $\mathcal{C}_{\mathbb{F}_q}$ is the **dual code** of $\mathcal{C}_{\mathbb{F}_q}$. The generator matrix of $\mathcal{C}_{\mathbb{F}_q}^{\perp}$ is the parity-check matrix of $\mathcal{C}_{\mathbb{F}_q}$. A code $\mathcal{C}_{\mathbb{F}_q}$ is said to be **self-dual** if $\mathcal{C}_{\mathbb{F}_q} = \mathcal{C}_{\mathbb{F}_q}^{\perp}$.

As a consequence, if G and H are the generator and the parity-check matrix, respectively, of a linear code $\mathcal{C}_{\mathbb{F}_q}$ over $\mathbb{F}_q$, then $HG^T = O$.

We say that a generator matrix G for an $[n,k]$ linear code is in **systematic form** if it has the form $G = [\ I_k \quad A\ ]$, where A has size $k \times (n-k)$. Respectively, a parity-check matrix H is in systematic form if it has the form $H = [\ B \quad I_{n-k}\ ]$, where B has size $(n-k) \times k$. In this case, if we have a codeword $\boldsymbol{c} = [\ c_0 \quad c_1 \quad \ldots \quad c_{n-1}\ ]$, we say that the first k symbols are the information symbols.

The next theorem establishes the relationship between the generator matrix and the parity-check matrix when they are in systematic form. The proof can be found in [24, 34], for example.

Theorem 1.3: *If $G = [\ I_k \quad A\ ]$ is a generator matrix for a linear code $\mathcal{C}_{\mathbb{F}_q}$ with parameters $[n,k]$ over $\mathbb{F}_q$ in systematic form, then $H = [\ -A^T \quad I_{(n-k)}\ ]$ is a parity-check matrix for $\mathcal{C}_{\mathbb{F}_q}$.*

1.2.3 Decoding

Let $\mathcal{C}_{\mathbb{F}_q}$ be a linear code with parameters $[n,k,d]$ over $\mathbb{F}_q$. The next theorem, whose proof can be found in [34] or [45], shows the relation between the minimum distance and the errors the code can correct.

Theorem 1.4: *An $[n, k, d]$ linear code can detect $d-1$ errors and correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.*

The higher the minimum distance is, the more errors the code can correct.

Suppose that the codeword $\boldsymbol{c}$ has been sent down a noisy channel and one or more of the codeword symbols may have been flipped. We receive the error-corrupted word $\widehat{\boldsymbol{c}} = \boldsymbol{c} + \boldsymbol{e}$ and the task of the decoder is to detect any flipped symbol and, if possible, to correct them. The most simple error decoding algorithm is called the **maximum likelihood** decoder. It chooses the codeword that is most likely to have been produced, that is, the closest codeword to $\widehat{\boldsymbol{c}}$, according to the Hamming distance. Let us write

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor, \tag{1.2}$$

if no more than t errors occurred, we would be able to find the closest codeword (see the next example).

Example 1.2: Consider the binary linear code given by

$$\mathcal{C}_{\mathbb{F}_2} = \{000000, 101010, 010101, 111111\}.$$

If we receive the word 100000, the codeword most likely to have been sent is 000000.■

Given the parity-check matrix H we define the vector of syndromes of $\widehat{\boldsymbol{c}}$ as $\boldsymbol{s}^T = H\widehat{\boldsymbol{c}}^T$. When $\widehat{\boldsymbol{c}}$ is a codeword, the vector $\boldsymbol{s}$ is the zero vector. In Section 2.6 and Section 3.5, we propose algorithms based on the vector of syndromes to decode the codes constructed in Chapters 2 and 3, respectively. Also, in Section 4.2 we propose an algorithm based on the syndromes to break an LFSR, considered as a code.

1.2.4 MDS Codes

An $[n, k, d]$ linear code over $\mathbb{F}_q$ contains q^k codewords. Then, if we replace $A_q(n, d)$ by q^k in expression (1.1), the Singleton bound for linear codes is given by (see [45], for instance)

$$d \leq n - k + 1. \tag{1.3}$$

A code attaining the Singleton bound is said to be a **maximum distance separable code** or MDS code for short. The following theorems provide some equivalent conditions for a code to be MDS. The proof can be found in [24] or [34], for example.

Theorem 1.5: *Let $\mathcal{C}_{\mathbb{F}_q}$ be an $[n,k]$ linear code over $\mathbb{F}_q$ with $k \geq 1$. Then the following statements are equivalent:*

(a) *$\mathcal{C}_{\mathbb{F}_q}$ is MDS.*
(b) *Every k columns of a generator matrix G are linearly independent (i.e. any k symbols of the codeword may be taken as message symbols).*
(c) *$\mathcal{C}_{\mathbb{F}_q}^{\perp}$ is MDS.*
(d) *Every $n-k$ columns of a parity-check matrix H are linearly independent.*

For binary codes, we have some extra properties.

Theorem 1.6: *Let $\mathcal{C}_{\mathbb{F}_2}$ be an $[n,k,d]$ linear binary code.*

(a) *If $\mathcal{C}_{\mathbb{F}_2}$ is MDS, then $\mathcal{C}_{\mathbb{F}_2}$ is trivial.*
(b) *If $d \geq 3$ and $k \geq 5$, then $k \leq n-d-1$.*

Remember that the trivial families of MDS linear codes are the full vector space whose parameters are $[n,n,1]$, repetition codes whose parameters are $[n,1,n]$ of any length $n \in \mathbb{N}$, and their duals.

1.2.5 Cyclicity

In this section we review an important property: the cyclicity.

Definition 1.6: Let $\mathcal{C}_{\mathbb{F}_q}$ be a linear code over $\mathbb{F}_q$ of length n. The code $\mathcal{C}_{\mathbb{F}_q}$ is called a **cyclic code**, if for $\boldsymbol{c} = [\ c_0 \ \ c_1 \ \ \cdots \ \ c_{n-2} \ \ c_{n-1}\ ] \in \mathcal{C}_{\mathbb{F}_q}$ the word $[\ c_{n-1} \ \ c_0 \ \ c_1 \ \ \cdots \ \ c_{n-2}\ ] \in \mathcal{C}_{\mathbb{F}_q}$.

That is, the word obtained by a cyclic right shift of the components of a codeword is again a codeword (see the next example).

Example 1.3: The binary code $\mathcal{C}_{\mathbb{F}_2}$ given in Example 1.2,

$$\mathcal{C}_{\mathbb{F}_2} = \{000000, 101010, 010101, 111111\},$$

is a cyclic code. ■

We continue with the classic definition of quasi-cyclic code.

Definition 1.7: A linear code $\mathcal{C}_{\mathbb{F}_q}$ of length $n = Nb$ over $\mathbb{F}_q$ is said to be a **quasi-cyclic code** of index b if for $\boldsymbol{c} = [\ c_0 \quad c_1 \quad \cdots \quad c_{n-2} \quad c_{n-1}\ ] \in \mathcal{C}_{\mathbb{F}_q}$ the word $[\ c_{n-b} \quad c_{n-b+1} \quad \cdots \quad c_{n-1} \quad c_0 \quad c_1 \quad \cdots \quad c_{n-b-1}\ ] \in \mathcal{C}_{\mathbb{F}_q}$.

This means that the word obtained by b cyclic shifts of the components of a codeword is also a codeword (see the next example).

Example 1.4: The linear binary code $\mathcal{C}_{\mathbb{F}_2}$ given by

$$\mathcal{C}_{\mathbb{F}_2} = \{000000, 111000, 000111, 111111\}$$

is quasi-cyclic of index 3. ■

1.2.6 LDPC Codes

Binary low-density parity-check codes (LDPC) were discovered by Gallager in the early 1960s [21]. However, they were forgotten until the late 1990s, when they were rediscovered. Nowadays these codes are widely studied [41, 53, 56, 59, 62]. These codes are characterized by having a sparse parity-check matrix, that is, the parity-check matrix contains mostly 0s and relatively few 1s. They can be divided into regular LDPC codes and irregular codes.

A binary (i, j)-**regular LDPC** code is defined as the null space of a sparse parity-check matrix over $\mathbb{F}_2$ such that:

(a) Each row contains j nonzero entries.
(b) Each column contains i nonzero entries.
(c) Any two columns have at most one nonzero entry in common.
(d) Both i and j are small compared with the length of the code.

An LDPC code is said to be **irregular** if its parity-check matrix has varying column weight or varying row weight.

1.3 Codes over Extension Alphabets

Let $\mathbb{F}_q$ be the Galois field of q elements and assume that $\mathcal{C}_{\mathbb{F}_q}$ denotes a linear code over $\mathbb{F}_q$. From now on, we consider $\mathrm{Mat}_{t\times m}(\mathbb{F}_q^b)$ the ring of matrices of size $t \times m$ with elements in $\mathbb{F}_q^b$.

1.3.1 Definition

Now, we recall the definition of $\mathbb{F}_q$-linear codes [7, 33].

Definition 1.8: Let b be a positive integer. A code $\mathcal{C}_{\mathbb{F}_q^b}$ is said to be an **$\mathbb{F}_q$-linear code** of length n over $\mathbb{F}_q^b$ if it is a linear subspace of the vector space $\mathbb{F}_q^{nb}$. We can also say that $\mathcal{C}_{\mathbb{F}_q^b}$ is an $\mathbb{F}_q$-linear code over $\mathbb{F}_q^b$ if the code $\mathcal{C}_{\mathbb{F}_q}$ is a linear code of length nb over $\mathbb{F}_q$. Therefore, the codewords of $\mathcal{C}_{\mathbb{F}_q^b}$ can be seen as codewords of length nb over $\mathbb{F}_q$.

Notice that both $\mathcal{C}_{\mathbb{F}_q}$ and $\mathcal{C}_{\mathbb{F}_q^b}$ refer to the same set of codewords, but considering the alphabet $\mathbb{F}_q$ and $\mathbb{F}_q^b$, respectively.

It is worth pointing out that the code symbols of $\mathcal{C}_{\mathbb{F}_q^b}$ can be regarded as elements in the field $\mathbb{F}_{q^b}$. However, linearity over this field is not assumed.

Example 1.5: Consider the following $\mathbb{F}_2$-linear code over $\mathbb{F}_2^2$ with length 2,

$$\mathcal{C}_{\mathbb{F}_2^2} = \{00\,00, 10\,01, 01\,10, 11\,11\}.$$

The alphabet in this case is $\mathbb{F}_2^2 = \{00, 01, 10, 11\}$. If we denote by $\boldsymbol{i}$ the binary expansion of 2 digits of the integer i, for $i = 0, 1, 2, 3$, then

$$\mathbb{F}_2^2 \approx \mathbb{F}_{2^2} = \{\boldsymbol{i} \mid i = 0, 1, 2, 3\}.$$

Then, the symbols of the alphabet can be also represented as $\mathbb{F}_{2^2} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Then, the code can be represented in the following way

$$\mathcal{C}_{\mathbb{F}_2^2} = \{\mathbf{00}, \mathbf{21}, \mathbf{12}, \mathbf{33}\}.$$ ■

However, the code is not a linear subspace over $\mathbb{F}_{2^2}$, since $\mathbf{2}(\mathbf{33}) = \mathbf{11} \notin \mathcal{C}_{\mathbb{F}_2^2}$.

The dual code $\mathcal{C}_{\mathbb{F}_q^b}^{\perp}$ is the null space of $\mathcal{C}_{\mathbb{F}_q^b}$ in $\mathbb{F}_q^b$.

Now, we analyze the relationship between the parameters of the code over $\mathbb{F}_q$ and the parameters of the code over $\mathbb{F}_q^b$. Let $[N, K, D]$ denote the parameters of the code $\mathcal{C}_{\mathbb{F}_q}$ over $\mathbb{F}_q$. The number $k = \log_{q^b} \left|\mathcal{C}_{\mathbb{F}_q^b}\right|$ is called the **normalized dimension** (or just dimension) of $\mathcal{C}_{\mathbb{F}_q^b}$ over $\mathbb{F}_q^b$. If b divides K then $k = K/b$ (in what follows, b divides K). Thus, the parameters of the code $\mathcal{C}_{\mathbb{F}_q^b}$ are $[n, k, d]$ over $\mathbb{F}_q^b$, where d is the minimum distance and $n = N/b$. To define the minimum (Hamming) distance of $\mathcal{C}_{\mathbb{F}_q^b}$ we consider it as a code over the alphabet $\mathbb{F}_q^b$. Then, the distance d is measured with respect to the symbols of $\mathbb{F}_q^b$ (see [7]).

It is worth remembering that the code $\mathcal{C}_{\mathbb{F}_q^b}$ can be specified by either its parity-check matrix H of size $(n-k)b \times nb$ or its generator matrix G of size $kb \times nb$, both over $\mathbb{F}_q$. From practical considerations, $\mathbb{F}_q$-linear codes are required to be systematic, that is, its parity-check (or generator) matrix has to be systematic. The matrix H (respectively, G) is said to be **systematic** if it contains the identity matrix of size $(n-k)b \times (n-k)b$ (respectively, $kb \times kb$).

The next example shows an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^2$ with the generator matrix in systematic form.

Example 1.6: Consider the generator matrix G given by

$$G = \left[\begin{array}{cccc:cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right].$$

This matrix is in systematic form and is a generator matrix of a code $\mathcal{C}_{\mathbb{F}_2}$ of length 8 and dimension 4 over $\mathbb{F}_2$.

We can compute the sixteen codewords of the code, and we can divide the bits of each codeword into groups of two elements. Then if we consider the codewords over the new alphabet $\mathbb{F}_2^2$, the length of each codeword is 4 and the normalized dimension of our new code $\mathcal{C}_{\mathbb{F}_2^2}$ is $4/2 = 2$. Then, the generator matrix can be seen as the block matrix

$$G = \left[\begin{array}{cc:cc:cc:cc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right].$$

■

If we consider the linear code $\mathcal{C}_{\mathbb{F}_2}$ whose generator matrix is the one in Example 1.6, then the length of the code is 8, the dimension is 4 and the minimum distance is 3 over $\mathbb{F}_2$. It is possible to check that the minimum distance of the code $\mathcal{C}_{\mathbb{F}_2^2}$ over $\mathbb{F}_2^2$ is also 3. However, in general, the minimum distance of the code over $\mathbb{F}_q$ and the minimum distance of the same code over $\mathbb{F}_q^b$ do not match up as we can see in the following example.

Example 1.7: Consider the $\mathbb{F}_2$-linear code with parameters $[5, 2]$ over $\mathbb{F}_2^2$ given by the following matrix

$$G = \left[\begin{array}{cc:cc:cc:cc:cc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right].$$

It is possible to check that although the minimum distance of the code $\mathcal{C}_{\mathbb{F}_2^2}$ over $\mathbb{F}_2^2$ is 2, the minimum distance of the code $\mathcal{C}_{\mathbb{F}_2}$ over $\mathbb{F}_2$ is 3. ■

In general, the minimum distance of $\mathcal{C}_{\mathbb{F}_q}$ over $\mathbb{F}_q$ is greater than or equal to the minimum distance of $\mathcal{C}_{\mathbb{F}_q^b}$ over $\mathbb{F}_q^b$, as we can see in the following result. We have not found the proof in any reference, so we show the proof below.

Theorem 1.7: *Let $\mathcal{C}_{\mathbb{F}_q^b}$ be an $\mathbb{F}_q$-linear code with parameters $[n, k]$ over $\mathbb{F}_q^b$. If d is the minimum distance of the code $\mathcal{C}_{\mathbb{F}_q^b}$ over $\mathbb{F}_q^b$ and D is the minimum distance of the code $\mathcal{C}_{\mathbb{F}_q}$ over $\mathbb{F}_q$, then*

$$d \leq D.$$

PROOF: Assume that $d > D$. Let $\boldsymbol{c} = [\ c_0 \quad c_1 \quad \ldots \quad c_{nb-1}\ ]$ be a codeword of the code $\mathcal{C}_{\mathbb{F}_q}$ over $\mathbb{F}_q$ with minimum weight, that is, $\omega(\boldsymbol{c}) = D$. Then, there exist D nonzero components of $\boldsymbol{c}$, say $c_{i_0}, c_{i_2}, \ldots, c_{i_{D-1}}$ and the remaining components are zero. If we divide $\boldsymbol{c}$ into groups of length b

$$\boldsymbol{c} = [\ c_0\ c_1 \ldots c_{b-1} \quad c_b\ c_{b+1} \ldots c_{2b-1} \quad \ldots \quad c_{b(n-1)}\ c_{b(n-1)+1} \ldots c_{bn-1}\ ],$$

then we can consider the codeword as a word over the alphabet $\mathbb{F}_q^b$. In this case, suppose we have each c_{i_j}, with $j \in \{0, 1, \ldots, D-1\}$ in a different group, then the

weight of the codeword over $\mathbb{F}_q^b$ is D, which is a contradiction, since the minimum distance (and, thus the minimum weight) over $\mathbb{F}_q^b$ is $d > D$. If we have more than one c_{i_j} in the same group, with $j \in \{0, 1, 2, \ldots, D-1\}$, then the weight of the codeword over $\mathbb{F}_q^b$ is even smaller than D, which is again a contradiction. □

It is worth pointing out that though this distance is greater it does not mean that the code is better. We cannot compare these two codes, since both alphabets are different. What we need is the code to be MDS (see Section 1.3.3), so if we have a code which is not MDS over $\mathbb{F}_q$ but it is MDS over $\mathbb{F}_q^b$ we do prefer to work over $\mathbb{F}_q^b$.

1.3.2 Relationship with Array Codes

Array codes are a sort of error and burst correcting codes, which have been studied by several authors [7, 33, 61]. In [1, 3, 6, 60, 61] we can find applications in storage systems and communication. Moreover, we can find applications of such codes in disk arrays in [7]. The array codes which are MDS are much more efficient in computational complexity terms than Reed-Solomon codes [6]. Furthermore, array codes provide the greatest protection against device errors, given an amount of redundancy. It is possible to find some constructions of this kind of codes in [1, 3, 7, 33, 60, 61]. In many cases array codes designed to correct burst error patterns can be both optimal (that is, MDS) and simpler to decode than other equivalent codes. Moreover, array codes are ideally suited for turbo decoding.

Apparently, there is no relationship between array codes and $\mathbb{F}_q$-linear codes, but in [5] we can find the explanation. The array codes to be considered are $b \times n$ arrays over $\mathbb{F}_2$. We may assume that k of the columns carry information and the remaining $n-k$ are parity columns. Thus, we can view the array as an $[n, k, d]$ $\mathbb{F}_2$-linear code over $\mathbb{F}_2^b$, where $\mathbb{F}_2^b$ represents the set of column vectors, and the minimum distance d is the minimal number of nonzero columns for a nonzero array in the code.

For example, the EVENODD code considered in [1, 5] was an $\mathbb{F}_2$-linear MDS code $\mathcal{C}_{\mathbb{F}_2^{p-1}}$ with parameters $[p+2, p, 3]$ over $\mathbb{F}_2^{p-1}$, with p a prime number. The parity-check matrix was given by

$$H_{EVODD} = \left[\begin{array}{c|c|c|c|c|c|c} I_{p-1} & I_{p-1} & I_{p-1} & \cdots & I_{p-1} & I_{p-1} & O_{p-1} \\ I_{p-1} & T_{p-1}^{(1)} & T_{p-1}^{(2)} & \cdots & T_{p-1}^{(p-1)} & O_{p-1} & I_{p-1} \end{array}\right],$$

where $T^{(l)} = \left[t_{i,j}^{(l)}\right]$, with

$$t_{i,j}^{(l)} = \begin{cases} 1, & \text{if} \quad j \neq p-l \quad \text{and} \quad j-i \equiv l \pmod{p}, \\ 1, & \text{if} \quad j = p-l, \\ 0, & \text{otherwise.} \end{cases}$$

The code can be also seen as an array code of size $(p-1) \times (p+2)$.

1.3.3 MDS Codes

In Section 1.2.4 the Singleton bound for linear codes was introduced in expression (1.3). The next theorem checks that the Singleton bound for $\mathbb{F}_q$-linear codes is the same.

Theorem 1.8: *For an $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[n,k,d]$ over $\mathbb{F}_q^b$, then*

$$d \leq n-k+1.$$

PROOF: We know that

$$M \leq (q^b)^{n-d+1}$$

where M is the number of codewords in $\mathcal{C}_{\mathbb{F}_q^b}$. Let the parameters of the code $\mathcal{C}_{\mathbb{F}_q}$ be $[N,K,D]$ over $\mathbb{F}_q$. Since $|\mathcal{C}_{\mathbb{F}_q}| = |\mathcal{C}_{\mathbb{F}_q^b}|$, we have that $M = q^K$. Then,

$$q^K \leq (q^b)^{n-d+1}$$

and using the properties of the logarithms we obtain

$$K \leq b(n-d+1).$$

Taking into account that $K = kb$, we have

$$k \leq n-d+1,$$

and the theorem holds. □

An $\mathbb{F}_q$-linear code with parameters $[n,k,d]$ is **MDS** over $\mathbb{F}_q^b$ if the Singleton bound given in Theorem 1.8 is attained. As we saw in Section 1.2.3 and Theorem 1.4 the correcting capacity of the codes depends on the minimum distance and the higher the minimum distance is, the more errors the code can correct.

Example 1.8: In example 1.6, it is possible to check that every codeword of the code $\mathcal{C}_{\mathbb{F}_2^2}$ has weight 3 or 4, considering the alphabet $\mathbb{F}_2^2$. Then, the minimum distance of the code is 3 over $\mathbb{F}_2^2$. Since the length is 4 and the dimension is 2, the code is an MDS $\mathbb{F}_2$-linear code over $\mathbb{F}_2^2$. ■

The two following theorems provide very useful characterizations in order to check whether an $\mathbb{F}_q$-linear code is MDS or not without computing the minimum distance [7]. These theorems are an extension of the characterization theorems for MDS block codes in [34]. The first theorem can be used when we have the parity-check matrix of the code. The second one allows us to check if the code is MDS when we have either the generator or the parity-check matrix in systematic form.

Theorem 1.9 ([7]): *Let $H = [\ H_0 \quad H_1 \quad \dots \quad H_{n-1}\ ]$ be an $(n-k)b \times nb$ parity-check matrix of an $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[n,k]$ over $\mathbb{F}_q^b$, where each H_i is an $(n-k)b \times b$ submatrix of H. Then $\mathcal{C}_{\mathbb{F}_q^b}$ is MDS if and only if the $(n-k)b$ columns of any $n-k$ distinct submatrices H_i form a linearly independent set over $\mathbb{F}_q$.*

Theorem 1.10 ([7]): *Let $H = [\ A \quad I_{(n-k)b}\ ]$ be an $(n-k)b \times nb$ systematic parity-check matrix of an $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[n,k]$ over $\mathbb{F}_q^b$ and write $A = [A_{i,j}] \in \mathrm{Mat}_{(n-k)b\times kb}(\mathbb{F}_q^b)$, where each $A_{i,j}$ is a $b \times b$ block submatrix of A. Then $\mathcal{C}_{\mathbb{F}_q^b}$ is MDS if and only if every square submatrix of A consisting of full blocks submatrices $A_{i,j}$ is nonsingular.*

The next example illustrates the idea given in the previous theorem.

Example 1.9: We consider the code $\mathcal{C}_{\mathbb{F}_2^2}$ with parameters $[4,2]$ over $\mathbb{F}_2^2$, whose parity-check matrix is

$$H = [\ A \quad I_4\ ] = \left[\begin{array}{cc:cc:cc:cc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Since the matrices

$$A_{1,1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{1,2} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{2,1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad A_{2,2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and A are nonsingular, according to Theorem 1.10, the code is MDS over $\mathbb{F}_2^2$. ■

1.3.4 Cyclicity

In this subsection we define the concept of cyclicity for $\mathbb{F}_q$-linear codes.

Definition 1.9: An $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ with length n over $\mathbb{F}_q^b$ is said to be **cyclic** if $[\ \boldsymbol{c}_0 \quad \boldsymbol{c}_1 \quad \ldots \quad \boldsymbol{c}_{n-1}\ ] \in \mathcal{C}_{\mathbb{F}_q^b}$ implies that $[\ \boldsymbol{c}_{n-1} \quad \boldsymbol{c}_0 \quad \boldsymbol{c}_1 \quad \ldots \quad \boldsymbol{c}_{n-2}\ ] \in \mathcal{C}_{\mathbb{F}_q^b}$.

It is worth pointing out that $\boldsymbol{c}_i \in \mathbb{F}_q^b$, for $i = 0, 1, \ldots, n-1$. Therefore, the code $\mathcal{C}_{\mathbb{F}_q^b}$ is cyclic if and only if $\mathcal{C}_{\mathbb{F}_q}$ is quasi-cyclic with cyclic shift of size b (see Definition 1.7).

Consider P the matrix which shifts b positions the columns of the matrix H. That is, if $H = [\ H_0 \quad H_1 \quad \ldots \quad H_{n-1}\ ]$, with H_i a block of size $(n-k)b \times b$, then $H \cdot P = [\ H_{n-1} \quad H_0 \quad H_1 \quad \ldots \quad H_{n-2}\ ]$. The cyclicity of the code is related to the following property, whose proof is straightforward.

Theorem 1.11: *An $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ over $\mathbb{F}_q^b$ with parity-check matrix H is cyclic if and only if there exists an invertible matrix L_P of size $(n-k)b \times (n-k)b$ such that $H \cdot P = L_P \cdot H$, where L_P shifts the rows of H $(n-k)b/n$ positions.*

The next example helps us to understand the previous theorem.

Example 1.10: Consider the parity-check matrix of the code $\mathcal{C}_{\mathbb{F}_2^2}$ over $\mathbb{F}_2^2$ given by

$$H = \left[\begin{array}{cc:cc:cc:cc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right].$$

If we shift each block column to the left one position, wrapping the first block column around to the last position, we obtain the following matrix

$$H \cdot P = \left[\begin{array}{cc:cc:cc:cc} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] = L_P \cdot H$$

where

$$P = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad L_P = \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right].$$

As we can see, the rows are the same, but shifted one position down and the last row becomes the first one. Then, the code is cyclic. ■

1.4 LFSR

As we saw in Section 1.1, in symmetric cryptography, the task of generating a random looking keystream $\boldsymbol{r} = (r_0, r_1, r_2, \ldots)$ from a short key has to be solved. Algorithms which generate keystreams from a secret key are called **keystream generators**. It is possible to do this through some nonlinear recurrence relation

$$r_{t+l} = f(r_{t+l-1}, r_{t+l-2}, \ldots, r_t), \quad t = 0, 1, \ldots$$

having the initial state $(r_0, r_1, \ldots, r_{l-1})$ as the key. This recurrence sequence will act as a pseudo-random generator. If the recurrence relation is linear, then it is possible to implement the recurrence relation with a **linear feedback shift register** (LFSR) [18, 19, 58].

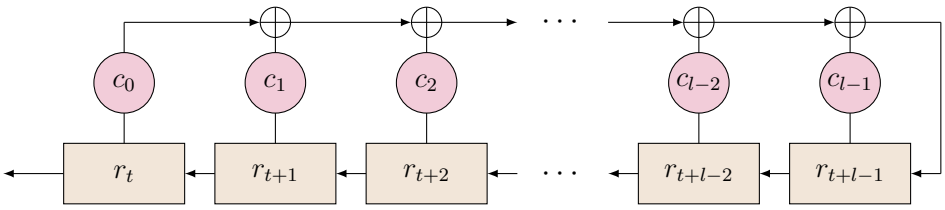


Figure 1.3: *An LFSR of length l.*

An LFSR of length l is a device consisting of l registers, called **taps**. Each tap is able to contain one symbol in each round. These symbols are elements from a field $\mathbb{F}_q$; in stream cipher applications we often have $q = 2$, the binary field, or some extension of the binary field $q = 2^m$, where m is the symbol size of the stream cipher. Thus, we will work in the finite field $\mathbb{F}_2$.

Each tap has a **feedback coefficient** $c_i \in \mathbb{F}_2$. It decides whether the bit in the according tap is used or not in the computation of the feedback sum. The taps with feedback coefficient equal to 1 are called **feedback taps**. In each round, the elements in the registers are shifted to the left and the element in the leftmost tap is the output of the LFSR in that round. The rightmost tap receives the sum of the bits in the feedback taps of the previous round, see Figure 1.3.

We assume the first feedback coefficient is $c_0 = 1$, otherwise one bit depends only on the $l - 1$ preceding bits. Hence the LFSR would have length $l - 1$. As we do not need more registers than necessary to make the feedback connection work, we also assume $c_l = 1$.

Let $\boldsymbol{r} = (r_0, r_1, r_2, \ldots)$ be an output sequence of the LFSR, then $\boldsymbol{r}$ fulfills the linear recurrence relation

$$c_0 r_t + c_1 r_{t+1} + \cdots + c_{l-1} r_{t+l-1} + r_{t+l} = 0, \quad t \geq 0.$$

The polynomial

$$f(\mathsf{x}) = c_0 + c_1 \mathsf{x} + \cdots + c_{l-1} \mathsf{x}^{l-1} + \mathsf{x}^l \in \mathbb{F}_2[\mathsf{x}]$$

is called the **feedback polynomial** of the LFSR. This polynomial gives a compact description of the LFSR. The sequence $\boldsymbol{r}$ is completely determined by its l initial bits $\boldsymbol{R}_0 = (r_0, r_1, \ldots, r_{l-1})$, called **initial state**. There exist 2^l different initial states, so the LFSR can generate 2^l different sequences. Note that one of the sequences is the

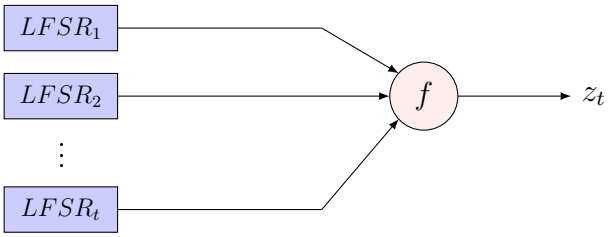


Figure 1.4: *The output of t LFSRs are combined via the nonlinear Boolean function f.*

zero sequence. We want to hide long messages, so we need the sequences to be as long as possible. The period τ of an LFSR with a primitive polynomial $f(\mathsf{x}) \in \mathbb{F}_2[\mathsf{x}]$ of degree l as the feedback polynomial and with a nonzero initial state, is $\tau = 2^l - 1$, the maximum possible period [31].

We recall the definition of linear complexity for a given sequence.

Definition 1.10: Given a sequence $\boldsymbol{s} = (s_0, s_1, \ldots, s_{N-1})$, we define the **linear complexity** of $\boldsymbol{s}$, denoted by $\mathcal{L}(\boldsymbol{s})$, as the the length of the shortest LFSR that generates the sequence.

The linear complexity can be determined with the Berlekamp-Massey algorithm [35], which efficiently computes the feedback polynomial of the LFSR given at least $2\mathcal{L}(s)$ of the output bits. Because of this, an LFSR is not a good keystream generator. The sequences generated have good statistical properties, desirable for keystream generator constructions, but we need to destroy the linearity, i.e. increase the linear complexity, before the sequence can be used. One classical approach is to use several binary LFSRs and combine the output from each one using a nonlinear Boolean function f as pictured in Figure 1.4. Remember that a **Boolean function** of n variables is a map $f : \mathbb{F}_2^n \to \mathbb{F}_2$, where n is a positive integer.

The properties of the output sequence will be good depending on the choice of the function f. Correlation attacks are based on the poor choice of this [16].

In the next example we show the well-known Geffe generator.

Example 1.11: The Geffe generator is a well-known nonlinear stream cipher. It consists of three LFSRs with different periods. In each step, the output of the

a_t	b_t	c_t	z_t
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Table 1.1: *Example of Geffe generator.*

generator equals the output of the first or the second LFSR, depending on the output of the third LFSR. Let a_t, b_t and c_t be the outputs of the different LFSRs at time t. The output z_t of the generator is defined by the following rule

$$z_t = \begin{cases} a_t, & \text{if} \quad c_t = 0, \\ b_t, & \text{if} \quad c_t = 1. \end{cases}$$

It is possible to check that for the Geffe generator the nonlinear Boolean function used to destroy the linearity is $f : \mathbb{F}_2^3 \to \mathbb{F}_2$, such that

$$f(x_1, x_2, x_3) = x_1(1 + x_3) + x_2 x_3.$$

All the possible cases are shown in Table 1.1. ■

In Chapter 4 we try to study LFSRs from the systems theory point of view. We can consider an LFSR as a code, and we try to recover the initial state using a decoding algorithm. We also introduce the idea of *generalized LFSR*.

1.5 Correlation Attacks

Correlation attacks are a class of plaintext attacks for breaking stream ciphers whose keystream is generated by combining the output of several LFSRs using a Boolean function as we saw in Figure 1.4. Correlation attacks exploit a statistical weakness that arises from a poor choice of the Boolean function.

1.5.1 Idea

As an illustration of the idea of correlation attack, we consider Example 1.11. In this example, we consider the Geffe generator. Remember that we had three input sequences from three different LFSRs a_t, b_t and c_t, and the output of the generator in each round is a_t or b_t, depending on the value of c_t. There are eight possible values for the outputs of the three registers and the value of the output of the combining function for each of them is shown in Table 1.1. Let us consider the output of the first register. One notices that the sequence $\boldsymbol{z}$ agrees in six of eight bits with the sequence $\boldsymbol{a}$, i.e., $f(a_t, b_t, c_t) = a_t$ in 75% of the cases. This generator is a good candidate to be attacked by a correlation attack.

Suppose we intercept the ciphertext $\boldsymbol{y}$ and we know n bits of the plaintext $\boldsymbol{m}$. It is possible to obtain n bits of the keystream just adding modulo 2 the known bits of $\boldsymbol{m}$ with the corresponding bits of $\boldsymbol{y}$. Now we may begin a brute force search of the space of possible keys (initial values) for the first LFSR. For any initial state, we compute the n corresponding bits of $\boldsymbol{a}$ and compare these to the n recovered bits of the keystream. We have established that there is a correlation of 75% between the output of the first LFSR $\boldsymbol{a}$ and the keystream. Then we know that if we have correctly guessed the key, approximately $3n/4$ bits will match. If we have guessed incorrectly, we expect $n/2$ to match. Thus we may recover the key for the first LFSR independently of the keys of the other LFSRs.

At this point we have reduced the problem of brute forcing a system of three LFSRs to the problem of brute forcing a single LFSR and then a system of two LFSRs. Observe in Table 1.1 that $\boldsymbol{b}$ also agrees with the keystream in 75% of the cases, so we can do the same with sequence $\boldsymbol{b}$.

Geffe generator is a very efficient keystream generator, but because of the correlation attacks it is not used any more.

This sort of correlation attacks was proposed by Siegenthaler [54, 55]. They suppose a significant reduction in complexity compared to the brute force approach. Suppose we have n LFSRs of length $l_0, l_1, \ldots, l_{n-1}$. In order to test every possible combination of initial states,

$$\prod_{i=0}^{n-1}(2^{l_i} - 1)$$

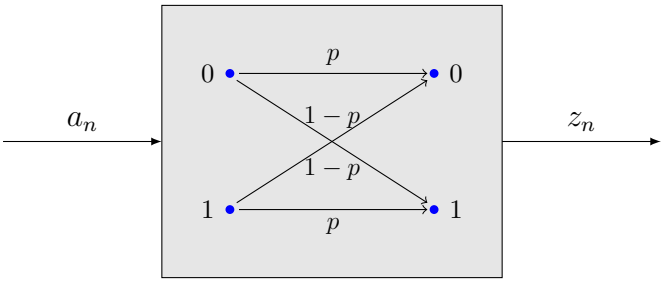


Figure 1.5: *BSC with crossover probability* $1-p$

different combinations have to be tested. Suppose that we have only one LFSR, the first one, concerned with the correlation attack, then the complexity is reduced by a factor 2^{l_0}

$$2^{l_0} + \prod_{i=1}^{n-1}(2^{l_i} - 1).$$

1.5.2 Fast Correlation Attacks

In [37, 38] Meier and Staffelbach proposed the first fast correlation attack. Let the output sequence $\boldsymbol{z} = (z_0, z_1, \ldots, z_{N-1})$ of a keystream generator be correlated to the output of an LFSR $\boldsymbol{a} = (a_0, a_1, \ldots, a_{N-1})$ of length l with correlation probability p higher than 0.5. They presented two algorithms, A and B, to determine the initial digits of $\boldsymbol{a}$. These algorithms were only efficient when the number t of feedback taps was small, for example $t \leq 10$ when $0.5 \leq p \leq 0.75$. The idea was to view the sequence $\boldsymbol{z}$ as a perturbation of the sequence $\boldsymbol{a}$ by a binary symmetric memoryless noise source, with $\Pr(z_n = a_n) = p$ (see Figure 1.5). Every digit a_n satisfies several linear relations derived from the feedback polynomial, all of them involving t other digits of $\boldsymbol{a}$. By substituting the corresponding digits of $\boldsymbol{z}$ in these relations, they obtained equations for each digit z_n which may hold or not. To test if $a_n = z_n$ they counted the number of equations which hold for z_n. The more of these equations hold, the higher is the probability for z_n to agree with the guessed a_n.

From the coding theory point of view they used a linear block code model (see [58]). Let us consider the sequence $\boldsymbol{a}$ to be a codeword of a linear block code. Then $\boldsymbol{z}$ is the received word, after the transmission over a BSC with crossover probability $1-p$ (see Figure 1.5). If one correctly decode $\boldsymbol{z}$ then can know the initial

state of the LFSR. In [37, 38] the algorithm A can be seen as an one-step decoding algorithm and algorithm B is an iterative algorithm very similar to the belief propagation algorithm known from LDPC codes [21, 49].

Other fast correlations attacks and improvements have been proposed after Meier and Staffelbach, see for example [14, 18, 27, 40].

Chapter 2

Cyclic Low Density MDS Codes

2.1 Introduction

Let $\mathbb{F}_2$ be the Galois field of 2 elements. In this chapter we introduce the construction of $\mathbb{F}_2$-linear codes with good properties such that cyclicity, low density or maximum distance separation (it is possible to extend these results for $\mathbb{F}_q$ the Galois field of $q > 2$ elements). For this purpose, we consider a specific partition of $\mathbb{Z}_n$, with $n = p - 1$ and p a prime odd number. Then, we apply a Zech logarithm to the elements of these sets, constructing new sets. We construct an index array representing a matrix, using these new sets. This matrix is the parity-check matrix of an $\mathbb{F}_2$-linear cyclic code with low density. The MDS property needs some extra conditions. For this aim some definitions are required.

Sometimes we have to work with matrices with large size. It can be a hard task, due to the possibility of making mistakes. That is why we introduce the concept of index array. This array stores the positions of the nonzero elements.

Definition 2.1: Let $\mathcal{C}_{\mathbb{F}_2^b}$ be an $\mathbb{F}_2$-linear code with parameters $[n, k]$ over $\mathbb{F}_2^b$ and let $H = [h_{i,j}]$ be an $(n-k)b \times nb$ parity-check matrix over $\mathbb{F}_2$. The matrix H can be represented by a $b \times n$ array of sets. The cell in location (i, j) contains the set $\{t \mid h_{t,i+bj} = 1\}$. This array is called **index array** of H.

Basically, each cell represents each column of the parity-check matrix, and inside the cell we store the position of the 1s in the corresponding column. We often refer to an index array just with array.

Given a matrix of size $m \times n$ the numeration of rows (respectively, columns) starts with 0 and ends with $m-1$ (respectively, $n-1$). The following example clarifies this definition.

Example 2.1: We consider the linear code $\mathcal{C}_{\mathbb{F}_2}$ with parameters $[18, 12]$ over $\mathbb{F}_2$, whose parity-check matrix is given by

$$H = \left[\begin{array}{ccc:ccc:ccc:ccc:ccc:ccc} 1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1 \\ 0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&1&0&0&1&0&0&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&0 \\ 0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0 \\ 0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0 \end{array}\right].$$

Of course, this code can be also seen as an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^3}$ with parameters $[6, 4]$ over $\mathbb{F}_2^3$. We can represent the matrix H by the index array

$$A_H = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 4,5 & 5,0 & 0,1 & 1,2 & 2,3 & 3,4 \\ \hline 1,3 & 2,4 & 3,5 & 4,0 & 5,1 & 0,2 \\ \hline \end{array}$$

where each cell represents each column of H and each block column of the array represents each block of columns of H. For example, if we observe the column in red, we can see that there is one 1 in the third position and another 1 in the fifth position. Then, we have to write 3 and 5 in the corresponding cell (the one in red).■

Given the index array representing a matrix M, we can obtain directly the index array of the transpose matrix M^T. We suppose there are not zero rows in M in the last position. If the indices inside the index array A_M are $0, 1, 2, \ldots, n-1$, then we know the transpose matrix must have n columns and, therefore, the index array A_{M^T} have n cells. Inside of the ith cell of A_{M^T}, we have to write the position of the cells containing the index i in A_M. The next example shows us how to follow this process.

Example 2.2: Consider the following index array:

$$A_M = \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 0,1 & 0,2 \\ \hline \end{array}$$

We observe the indices in the cells of A_M are $0, 1$ or 2, then we suppose the matrix M^T has three columns and therefore, the array A_{M^T} has three cells. In the 0th cell we write the positions of 0 in A_M, in the 1st cell we write the positions of 1 and in the 2nd cell we write the positions of 2:

$$A_{M^T} = \boxed{0,1,3} \boxed{1,2} \boxed{3}.$$

The matrices M and M^T are

$$M = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix},$$

so we can check the index arrays are correct. ■

The construction in Section 2.3 makes use of the well-known Zech logarithm. We need to recall this concept.

Definition 2.2: Let α be a primitive element of $\mathbb{F}_p$, with p a prime integer. The **Zech logarithm** with base α can be defined as the map

$$\begin{array}{rccc} \mathcal{Z}_\alpha : & \mathbb{Z}_{p-1} & \longrightarrow & \mathbb{Z}_{p-1}^\star \cup \{\infty\} \\ & x & \rightsquigarrow & \mathcal{Z}_\alpha(x), \end{array}$$

where $\alpha^x + 1 = \alpha^{\mathcal{Z}_\alpha(x)}$. It is convenient to choose $\alpha^\infty = 0$.

In [23] we can find interesting properties of Zech logarithms.

2.2 Gauss-Jordan Elimination Algorithm

There are many ways to deal with the problem of knowing whether a matrix is nonsingular. For example, we can use the Gauss-Jordan elimination algorithm. If after several steps we obtain a triangular matrix, where the elements in the main diagonal are nonzero, or the identity matrix, then the matrix is nonsingular.

In case we have an index array representing a binary matrix, we can also apply this algorithm. In this case, when we "add" the sets cointained in two cells, the resultant cell contains all the elements in both sets but the repeated ones. If we add, for instance, a cell containing the set $\{1,2\}$ to another one containing the set $\{2,3\}$, the resultant one contains the new set $\{1,3\}$. If after several steps we get an empty cell (this represents a zero column in the matrix), then the matrix is singular.

Example 2.3: Consider the matrix M and the array A_M which represents it. We apply the Gauss-Jordan elimination method for M and, at the same time, we see the changes produced in the array.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longrightarrow A_M = \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1,2,3 & 3,4,5 \\ \hline 1,3,5 & 0,2,4 \\ \hline \end{array}$$

First of all, we use the 0th column in order to eliminate every 1 in the 0th row. In the array, we use the cell containing 0 to remove every 0 in other cells.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1,2,3 & 3,4,5 \\ \hline 1,3,5 & 0,2,4 \\ \hline \end{array}$$

Now, we use the 3rd column to remove every 1 in the 1st row. That is, we use the cell with 1 to eliminate every 1 in other cells.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1,2,3 & 3,4,5 \\ \hline 1,3,5 & 2,4 \\ \hline \end{array}$$

In order to eliminate the 1s in the 2nd row, we choose, for example, the 1st column. We have to be careful, since this column has another 1 in another position. Speaking in array terms, we use the set $\{2,3\}$ to remove every 2. We have to add the integer 3 in every cell containing 2. If there was already 3 in that cell we can remove it.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 2,3 & 3,4,5 \\ \hline 3,5 & 2,4 \\ \hline \end{array}$$

To eliminate the 1s in the 3rd row, we can choose the 2nd, the 4th or the 5th column. We choose, for instance, the 2nd column. In the array, we use the set $\{3,5\}$ in order to remove every 3 in the remaining cells. We have to add or remove the integer 5, as we did in last step.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 2,3 & 3,4,5 \\ \hline 3,5 & 3,4 \\ \hline \end{array}$$

We choose the 4th column to eliminate the 1s in the 4th row. In the array, we use the cell with the single 4 in order to eliminate every 4 in other cells.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 2,5 & 4 \\ \hline 3,5 & 4,5 \\ \hline \end{array}$$

Now, we use the last column to eliminate the 1s in the last row. In the array, we use the cell with 5 in order to eliminate every 5 in other cells.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 2,5 & 4 \\ \hline 3,5 & 5 \\ \hline \end{array}$$

If we put in order the columns, we have the identity matrix. As a consequence, we can say the matrix M is nonsingular

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{array}{|c|c|} \hline 0 & 3 \\ \hline 1 & 4 \\ \hline 2 & 5 \\ \hline \end{array}$$

■

2.3 Construction

We construct an index array which represents a parity-check matrix of an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^b$ with good properties.

Let p be an odd prime integer and let $n = p - 1$. Then we can choose two integers b and r such that $n = rb$. We construct an index array corresponding to the parity-check matrix H of an $\mathbb{F}_2$-linear code with parameters $[n, n - r]$ over $\mathbb{F}_2^b$.

First of all, we consider the following sets

$$E_i = \{x \in \mathbb{Z}_n \mid i = x \bmod b\}, \quad i = 0, 1, \ldots, b - 1, \tag{2.1}$$

which form a partition of $\mathbb{Z}_n$.

0	1	2	$\cdots$	$p-3$	$p-2$
D_0	D_0+1	D_0+2	$\cdots$	D_0+p-3	D_0+p-2
D_1	D_1+1	D_1+2	$\cdots$	D_1+p-3	D_1+p-2
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
D_{u-1}	$D_{u-1}+1$	$D_{u-1}+2$	$\cdots$	$D_{u-1}+p-3$	$D_{u-1}+p-2$
D_{u+1}	$D_{u+1}+1$	$D_{u+1}+2$	$\cdots$	$D_{u+1}+p-3$	$D_{u+1}+p-2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
D_{b-1}	$D_{b-1}+1$	$D_{b-1}+2$	$\cdots$	$D_{b-1}+p-3$	$D_{b-1}+p-2$

Table 2.1: *Index array constructed in Section 2.3.*

Let α be a primitive element of $\mathbb{F}_p$. Let us consider $\Phi : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n^\star \cup \{\infty\}$ the map given by $\Phi(x) = \mathcal{Z}_\alpha(x)$, where $\mathcal{Z}_\alpha(x)$ is the Zech logarithm of x in base α given in Definition 2.2. Since

$$\alpha^{\frac{n}{2}} + 1 = p - 1 + 1 = 0 \bmod p = \alpha^\infty$$

(see [23]) we have that $\Phi(\frac{n}{2}) = \infty$. Let $u = \frac{n}{2} \bmod b$. Then, it is easy to check that $\frac{n}{2} \in E_u$. We eliminate this set and we apply Φ to the other sets.

Denote $D_i = \Phi(E_i)$, for $i = 0, 1, \ldots, b-1$, $i \neq u$. The index array A_H representing the parity-check matrix H has the form given in Table 2.1.

From this index array we can obtain the parity-check matrix H of the code. Here we have an example of this construction.

Example 2.4: Let $p = 11$. Then $n = 10$ and, therefore we can choose $r = 2$ and $b = 5$. According to expression (2.1) we can construct the following partition of $\mathbb{Z}_{10}$,

$$E_0 = \{0, 5\},\ E_1 = \{1, 6\},\ E_2 = \{2, 7\},\ E_3 = \{3, 8\} \quad \text{and} \quad E_4 = \{4, 9\}.$$

Let $\alpha = 2$ be a primitive element of $\mathbb{F}_{11}$. Then we compute the following Zech logarithm table:

x	0	1	2	3	4	5	6	7	8	9
$\mathcal{Z}_\alpha(x)$	1	8	4	6	9	∞	5	3	2	7

0	1	2	3	4	5	6	7	8	9
5,8	6,9	7,0	8,1	9,2	0,3	1,4	2,5	3,6	4,7
3,4	4,5	5,6	6,7	7,8	8,9	9,0	0,1	1,2	2,3
2,6	3,7	4,8	5,9	6,0	7,1	8,2	9,3	0,4	1,5
7,9	8,0	9,1	0,2	1,3	2,4	3,5	4,6	5,7	6,8

■

Table 2.2: *Index array constructed in Example 2.4.*

In this case $u = 0 = 5 \bmod 5$, so we eliminate the set $E_0 = \{0, 5\}$. Now, applying the Zech logarithm to the remaining sets we obtain the following sets,

$$D_1 = \{5, 8\},\ D_2 = \{3, 4\},\ D_3 = \{2, 6\} \quad \text{and} \quad D_4 = \{7, 9\}.$$

These sets form the 0th column of the index array. The rest of the columns can be obtained as we saw in Table 2.1. Then, the index array is given in Table 2.2.

It is worth noticing that for $r = 2$, in the 0th block column of the index array given in Table 2.1, we have the indices $\left\{\mathcal{Z}_\alpha(k), \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right)\right\}$, for $k \in \mathbb{Z}^*_{\frac{n}{2}}$, in each cell apart from the one in the 0th position (there we have $\{0\}$). The next theorem states that the set of subtractions between these two indices for every cell is $\mathbb{Z}^*_{\frac{n}{2}}$.

Theorem 2.1: *Let α be a primitive element in $\mathbb{F}_p$, with p an odd prime and consider $n = p - 1$. Then the set $\mathbb{Z}^*_{\frac{n}{2}}$ coincides with the set*

$$\left\{\min\left\{\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right)\right) \bmod n, \left(\mathcal{Z}_\alpha\left(k + \frac{n}{2}\right) - \mathcal{Z}_\alpha(k)\right) \bmod n\right\} \mid k \in \mathbb{Z}^*_{\frac{n}{2}}\right\}. \tag{2.2}$$

PROOF: First, we have to check that the numbers

$$\min\left\{\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right)\right) \bmod n, \left(\mathcal{Z}_\alpha\left(k + \frac{n}{2}\right) - \mathcal{Z}_\alpha(k)\right) \bmod n\right\},$$

for $k \in \mathbb{Z}^*_{\frac{n}{2}}$ are pairwise different.

We start with $\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right)$. Note that:

$$\alpha^{\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)} = \alpha^{\mathcal{Z}_\alpha(k)}\left(\alpha^{\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)}\right)^{-1}$$

$$= \left(1+\alpha^{k}\right)\left(1+\alpha^{k+\frac{n}{2}}\right)^{-1} = \left(1+\alpha^{k}\right)\left(1-\alpha^{k}\right)^{-1}$$

It is worth noticing that neither $\left(1-\alpha^{k}\right)$ nor $\left(1+\alpha^{k}\right)$ is zero, since $k \neq 0$ and $k \neq \frac{n}{2}$, respectively.

So, if we have,

$$\left(1+\alpha^{k_1}\right)\left(1-\alpha^{k_1}\right)^{-1} = \left(1+\alpha^{k_2}\right)\left(1-\alpha^{k_2}\right)^{-1}, \quad \text{for } k_1, k_2 \in \mathbb{Z}_{\frac{n}{2}}^{*}$$

then $\alpha^{k_1} = \alpha^{k_2}$. Since α is a primitive element, we obtain $k_1 = k_2$. That means the numbers $\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right) \bmod n$, for $k \in \mathbb{Z}_{\frac{n}{2}}^{*}$, are pairwise different.

Using the same argument, we can also check that the numbers $(\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) - \mathcal{Z}_\alpha(k)) \bmod n$, for different elements in $k \in \mathbb{Z}_{\frac{n}{2}}^{*}$, are pairwise different and the numbers $(\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) - \mathcal{Z}_\alpha(k)) \bmod n$ and $(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)) \bmod n$, for different elements in $k \in \mathbb{Z}_{\frac{n}{2}}^{*}$ are pairwise different.

Moreover $\mathcal{Z}_\alpha(k), \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) \in \mathbb{Z}_n$ and

$$\left[\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right) \bmod n\right] + \left[\left(\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) - \mathcal{Z}_\alpha(k)\right) \bmod n\right] = n.$$

Then we deduce that

$$\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right) \bmod n \leq \frac{n}{2} \quad \text{and} \quad \left(\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) - \mathcal{Z}_\alpha(k)\right) \bmod n \geq \frac{n}{2}$$

or

$$\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right) \bmod n \geq \frac{n}{2} \quad \text{and} \quad \left(\mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) - \mathcal{Z}_\alpha(k)\right) \bmod n \leq \frac{n}{2}.$$

Assume that $\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right) \bmod n = \frac{n}{2}$ (we can use the same argument for the other cases). Then, for some integer q, we have

$$\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) = qn + \frac{n}{2}. \tag{2.3}$$

On the other hand, we know that

$$0 \leq \mathcal{Z}_\alpha(k) \leq n \quad \text{and} \quad 0 \leq \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right) \leq n,$$

and, consequently,

$$\left|\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)\right| \leq n.$$

Therefore, in expression (2.3), $q = 0$ or $q = -1$.

Suppose that $\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right) = \frac{n}{2}$ (we can use the same argument with $\left(\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k + \frac{n}{2}\right)\right) = -\frac{n}{2}$). Then, we have

$$\alpha^{\mathcal{Z}_\alpha(k) - \mathcal{Z}_\alpha\left(k+\frac{n}{2}\right)} = \alpha^{\frac{n}{2}},$$

or, in other terms,

$$(1 + \alpha^k)(1 - \alpha^k)^{-1} = \alpha^{\frac{n}{2}}.$$

Now we add 1 to both members of the previous equation,

$$(1 + \alpha^k)(1 - \alpha^k)^{-1} + 1 = \alpha^{\frac{n}{2}} + 1.$$

Since the second member is 0 we have

$$(1 + \alpha^k) = (\alpha^k - 1),$$

and, then, we obtain that $2 = 0$, which is not possible.

Now, we know that elements in the set given in expression (2.2) are pairwise different and take values between 1 and $\frac{n}{2} - 1$. Therefore, this set is $\mathbb{Z}^*_{\frac{n}{2}}$ and the theorem holds. □

2.4 Construction Properties

Our construction has good properties, such as cyclicity, low density or the MDS property in some cases. In this section we proceed with the study of these properties.

2.4.1 MDS

In general, codes constructed in Section 2.3 are not MDS. In Table 2.3 we show for several values of p, r and α whether the constructed $\mathbb{F}_2$-linear code is MDS or not. As we can observe, the code is not always MDS.

The codes seem to be MDS when $r = 2$, no matter the primitive element. In this case, $r = 2$ and $b = \frac{n}{2}$, so the constructed code is an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^{\frac{n}{2}}$ with parameters $[n, n-2]$.

In general, when we have the index array instead of the parity-check matrix, we can write Theorem 1.9 in the following way.

		α	
		2	3
r	2	Yes	Yes

(a) $p = 5$

		α	
		3	5
r	2	Yes	Yes
	3	No	No

(b) $p = 7$

		α			
		2	6	7	8
r	2	Yes	Yes	Yes	Yes
	5	No	No	No	No

(c) $p = 11$

		α			
		2	6	7	11
r	2	Yes	Yes	Yes	Yes
	3	Yes	Yes	Yes	Yes
	4	No	No	No	No
	6	No	No	No	No

(d) $p = 13$

		α							
		3	5	6	7	10	11	12	14
r	2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	4	No	No	No	No	No	No	No	No
	8	No	No	No	No	No	No	No	No

(e) $p = 17$

		α					
		2	3	10	13	14	15
r	2	Yes	Yes	Yes	Yes	Yes	Yes
	3	Yes	Yes	Yes	Yes	Yes	Yes
	6	No	No	No	No	No	No
	9	No	No	No	No	No	No

(f) $p = 19$

Table 2.3: *MDS property for different values of p, r and α.*

Theorem 2.2: *Let A_H be a $b \times n$ index array representing the parity-check matrix H of an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^b}$ with parameters $[n, k]$ over $\mathbb{F}_q^b$. Then $\mathcal{C}_{\mathbb{F}_2^b}$ is an MDS code if and only if every $n-k$ columns of the index array represent a linearly independent set of columns of H over $\mathbb{F}_2$.*

According to Theorem 2.2, for $r = 2$ we select every 2 columns of the index array in Table 2.1, we apply the Gauss-Jordan elimination algorithm given in Section 2.2 and we obtain a subarray which represents the identity matrix of order n.

Example 2.5: In Example 2.1 we had the following index array:

$$A_H = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 4,5 & 5,0 & 0,1 & 1,2 & 2,3 & 3,4 \\ \hline 1,3 & 2,4 & 3,5 & 4,0 & 5,1 & 0,2 \\ \hline \end{array}$$

We select the first two columns of the index array and we apply the Gauss-Jordan elimination algorithm shown in Section 2.2.

$$\begin{array}{|c|c|} \hline 0 & 1 \\ \hline 4,5 & 5,0 \\ \hline 1,3 & 2,4 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 4,5 & 5 \\ \hline 1,3 & 2,4 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 4,5 & 5 \\ \hline 3 & 2,4 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 4 & 5 \\ \hline 3 & 2,4 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 4 & 5 \\ \hline 3 & 2 \\ \hline \end{array}$$

The last subarray represents the columns of the identity matrix of size 6×6. If we check for every pair of columns of the index array and we follow the same process, we obtain the same result. As a consequence, we can say the $\mathbb{F}_2$-linear code whose parity-check matrix is represented by this array is MDS over $\mathbb{F}_2^3$. ■

The codes constructed in Section 2.3 seem to be MDS when $r = 2$. As we saw in Theorem 2.1, the sets in the 0th column of the index array have a special form. Therefore, the MDS property must follow from this form of the cells in the index array. However, we have not found an easy proof yet.

2.4.2 Cyclicity

In this section, we prove that the codes we obtained in Section 2.3 are cyclic. For this aim, we recall some results. From Theorem 1.11 we can obtain the following corollary, whose proof is straightforward.

Corollary 2.1: *Let $\mathcal{C}_{\mathbb{F}_2^b}$ be an $\mathbb{F}_2$-linear code with parameters $[n, k]$ over $\mathbb{F}_2^b$. Let A_H be the index array which represents the parity-check matrix of the code $\mathcal{C}_{\mathbb{F}_2^b}$. $\mathcal{C}_{\mathbb{F}_2^b}$ is cyclic if and only if when we add $(n-k)b/n$ modulo n to each element of every set of the array, the resultant array represents a cyclic shift on the columns of A_H.*

The next example shows the idea given in the previous theorem.

Example 2.6: Consider the index array in Example 2.1. This array represents the parity-check matrix of an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^3$ with parameters $n = 6$ and $k = 4$ (with $b = 3$ and $r = 2$). Therefore, if we add 1 modulo n to every element in every set of the array A_H of Example 2.1 we obtain the following array

$0+1$	$1+1$	$2+1$	$3+1$	$4+1$	$5+1$
$4+1, 5+1$	$5+1, 0+1$	$0+1, 1+1$	$1+1, 2+1$	$2+1, 3+1$	$3+1, 4+1$
$1+1, 3+1$	$2+1, 4+1$	$3+1, 5+1$	$4+1, 0+1$	$5+1, 1+1$	$0+1, 2+1$

that is:

1	2	3	4	5	0
$5, 0$	$0, 1$	$1, 2$	$2, 3$	$3, 4$	$4, 5$
$2, 4$	$3, 5$	$4, 0$	$5, 1$	$0, 2$	$1, 3$

which represents the matrix

$$\left[\begin{array}{ccc:ccc:ccc:ccc:ccc:ccc} 0&1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0 \\ 1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&1 \\ 0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&1&0&0&1&0&0&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&1&0 \\ 0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0 \end{array}\right].$$

This matrix is obtained exchanging the 0th block and the final block of the parity-check matrix in Example 2.1. According to Corollary 2.1 the code is cyclic. ■

In our case $n = br$, then we have to add 1 to the elements of each set of the array (modulo n). If we observe the form of the array given in Table 2.1 and we add 1 to the elements in the sets, the 0th column is now the last column as we can see in Table 2.4.

Considering this fact and Corollary 2.1, we can say that these codes are cyclic.

1	2	$\cdots$	$p-2$	0
D_0+1	D_0+2	$\cdots$	D_0+p-2	D_0
D_1+1	D_1+2	$\cdots$	D_1+p-2	D_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$D_{u-1}+1$	$D_{u-1}+2$	$\cdots$	$D_{u-1}+p-2$	D_{u-1}
$D_{u+1}+1$	$D_{u+1}+2$	$\cdots$	$D_{u+1}+p-2$	D_{u+1}
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$D_{b-1}+1$	$D_{b-1}+2$	$\cdots$	$D_{b-1}+p-2$	D_{b-1}

Table 2.4: *Index array obtained adding* 1 *modulo* n *to every set in the array given in Table 2.1.*

2.4.3 Low Density

Blaum, Brady, Bruck and Menon [1] dealt with the problem of constructing $\mathbb{F}_2$-linear MDS codes with sparse (low density) systematic parity-check matrix. In this paper they presented the EVENODD code. As we saw in Subsection 1.3.2, this code was an MDS code over $\mathbb{F}_2^{m-1}$ with parameters $[m+2, m, 3]$, where m is an odd prime. Besides, the average number of 1s in a row of the generator matrix was equal to $4-(2/m)$.

On the other hand, the construction we present in this chapter yields $\mathbb{F}_2$-linear codes over $\mathbb{F}_2^b$ with parameters $[n, n-r]$, with $b=\frac{n}{r}$.

The number of 1s in each row of the parity-check matrix is $n-r+1$ (every integer in $\mathbb{Z}_n$, representing each row, appears $n-r+1$ times in the index array). On the other hand, we can control the number of 1s we have in each column. There are n columns containing one 1 and the number of 1s contained in the remaining $n(b-1)$ columns is r. Therefore, the number of 1s the matrix contains depends on the choice of r.

Example 2.7: Consider the array in Example 2.4. In this case $n=10$, $b=5$ and $r=2$. We can observe that every integer in $\mathbb{Z}_{10}$ appears $n-r+1=9$ times in the array. It means that each row of the matrix contains nine 1s.

On the other hand, the matrix has $n=10$ columns with one 1 and the number of 1s contained in the remaining $n(b-1)=40$ columns is 2. Therefore, the number of 1s in the matrix is 90. ■

0	1	2	$\cdots$	$p-3$	$p-2$
D_1	D_1+1	D_1+2	$\cdots$	D_1+p-3	D_1+p-2
D_2	D_2+1	D_2+2	$\cdots$	D_2+p-3	D_2+p-2
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
D_{b-1}	$D_{b-1}+1$	$D_{b-1}+2$	$\cdots$	$D_{b-1}+p-3$	$D_{b-1}+p-2$

Table 2.5: *Index array representing the parity-check matrix when* $r=2$.

In general, the average number of 1s in the matrix is

$$\frac{n(n-r+1)}{n^2b}=\frac{nr(n-r+1)}{n^3}.$$

We consider $r=2$, that is when the code is MDS. Then the average is

$$\frac{2n^2-2n}{n^3}.$$

When n is very large, the average is close to 0.

2.5 Computing the Generator Matrix

We consider the case $r=2$, that is when the code is MDS. In this case, the set E_0 in expression (2.1) is the one containing the element $u=\frac{n}{2} \bmod b$. Therefore, the index array representing the parity-check matrix of the code constructed in Section 2.3, is given by Table 2.5. The next theorem, shows us how to compute the index array of the generator matrix from the index array of the corresponding parity-check matrix.

Theorem 2.3: *Given the index array in Table 2.5, the index array representing the corresponding generator matrix is given in Table 2.6. For* $i=0,1,\ldots,n-1$ *and* $k=0,1,\ldots,b-1$, F_i *is the set containing the indices,*

$$\begin{cases}[j(b-1)+k-1]b, & \text{if } j=0,1 \text{ and } i\in D_k+j,\\ (j-2)b+k, & \text{if } j=2,3,\ldots,n-1 \text{ and } i\in D_k+j,\\ \emptyset, & \text{otherwise.}\end{cases}$$

PROOF: Given the index array in Table 2.5, if we move some cells, we obtain the array given in Table 2.7 (remember that in this case $b=\frac{n}{2}$ and $n=p-1$).

F_0	F_1	F_2	F_3	F_4	$\cdots$	F_{n-1}
0	$(b-1)b$	1	$b+1$	$2b+1$	$\cdots$	$(2b-3)b+1$
b	bb	2	$b+2$	$2b+2$	$\cdots$	$(2b-3)b+2$
$2b$	$(b+1)b$	3	$b+3$	$2b+3$	$\cdots$	$(2b-3)b+3$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$		$\vdots$
$(b-2)b$	$(2b-3)b$	$b-1$	$2b-1$	$3b-1$	$\ldots$	$(2b-2)b-1$

Table 2.6: *Index array representing the generator matrix computed from the index array in Table 2.5.*

$A_{H_s} =$

0	b	D_1	D_2	$\cdots$	D_{b-1}	D_1+1	D_2+1	$\cdots$	$D_{b-1}+1$
1	$b+1$	D_1+2	D_1+3	$\cdots$	D_1+b	D_1+b+1	D_1+b+2	$\cdots$	D_1+2b-1
2	$b+2$	D_2+2	D_2+3	$\cdots$	D_2+b	D_2+b+1	D_2+b+2	$\cdots$	D_2+2b-1
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
$b-1$	$2b-1$	$D_{b-1}+2$	$D_{b-1}+3$	$\cdots$	$D_{b-1}+b$	$D_{b-1}+b+1$	$D_{b-1}+b+2$	$\cdots$	$D_{b-1}+2b-1$

Table 2.7: *Index array representing the parity-check matrix in systematic form.*

This array represents the parity-check matrix in systematic form. In order to obtain the matrix in systematic form we have applied the permutation matrix, whose index array is given in Table 2.8. This matrix P indicates the new position of the columns in the index array. The corresponding index array representing the transpose matrix of P is given in Table 2.9. The array representing the generator matrix in systematic form is given in Table 2.10, where F_i is the set containing the positions of the cells containing i in the subarray of A_{H_s} given in Table 2.11. If we apply the permutation matrix P^T we obtain the index array representing the corresponding generator matrix given in Table 2.6. □

The next example shows how to obtain the index array representing the generator matrix from the index array representing the parity-check matrix.

Example 2.8: The index array in Example 2.1 has the form in Table 2.5. In this case

$A_P =$

0	b^2	1	2	$\cdots$	$b-1$	$b+1$	$\cdots$	$2b-1$
b	$(b+1)b$	$2b+1$	$3b+1$	$\cdots$	b^2+1	$(b+1)b+1$	$\cdots$	$(2b-1)b+1$
$2b$	$(b+2)b$	$2b+2$	$3b+2$	$\cdots$	b^2+2	$(b+1)b+2$	$\cdots$	$(2b-1)b+2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
$(b-1)b$	$(2b-1)b$	$3b-1$	$4b-1$	$\cdots$	$(b+1)b-1$	$(b+2)b-1$	$\cdots$	$2b^2-1$

Table 2.8: *Index array representing the permutation matrix P.*

$A_{P^T} =$

0	1	2	3	$\cdots$	$2b-1$
$2b$	$(b+1)b$	$2b+1$	$3b+1$	$\cdots$	$(2b-1)b+1$
$3b$	$(b+2)b$	$2b+2$	$3b+2$	$\cdots$	$(2b-1)b+2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
b^2	$(2b-1)b$	$3b-1$	$4b-1$	$\cdots$	$2b^2-1$

Table 2.9: *Index array representing the matrix P^T.*

$A_{G_s} =$

F_0	F_b	0	b	$\cdots$	$(2b-3)b$
F_1	F_{b+1}	1	$b+1$	$\cdots$	$(2b-3)b+1$
F_2	F_{b+2}	2	$b+2$	$\cdots$	$(2b-3)b+2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
F_{b-1}	F_{2b-1}	$b-1$	$2b-1$	$\cdots$	$(2b-2)b-1$

Table 2.10: *Index array representing the generator matrix in systematic form.*

$n = 6$ and $b = 3$. The array representing the generator matrix is given by:

F_0	F_1	F_2	F_3	F_4	F_5
0	6	1	4	7	10
3	9	2	5	8	11

D_1	D_2	$\dots$	D_{b-1}	D_1+1	D_2+1	$\cdots$	$D_{b-1}+1$
D_1+2	D_1+3	$\dots$	D_1+b	D_1+b+1	D_1+b+2	$\cdots$	D_1+2b-1
D_2+2	D_2+3	$\dots$	D_2+b	D_2+b+1	D_2+b+2	$\cdots$	D_2+2b-1
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
$D_{b-1}+2$	$D_{b-1}+3$	$\dots$	$D_{b-1}+b$	$D_{b-1}+b+1$	$D_{b-1}+b+2$	$\cdots$	$D_{b-1}+2b-1$

Table 2.11: *Subarray of the array given in Table 2.7.*

We construct the sets F_i, for $i = 0, 1, \dots, n-1$ as in Theorem 2.3. For $i = 0$, we can compute the elements in F_0 in the following way:

$$\begin{aligned}
0 \in D_1 + 1 \quad (k = 1, j = 1) &\rightarrow [1(3-1) + 1 - 1]3 = 6 \in F_0 \\
0 \in D_1 + 2 \quad (k = 1, j = 2) &\rightarrow (2-2)3 + 1 = 1 \in F_0 \\
0 \in D_2 + 3 \quad (k = 2, j = 3) &\rightarrow (3-2)3 + 2 = 5 \in F_0 \\
0 \in D_2 + 5 \quad (k = 2, j = 5) &\rightarrow (5-2)3 + 2 = 11 \in F_0
\end{aligned}$$

Now, for $i = 1$, the elements of the set F_1 are given by:

$$\begin{aligned}
1 \in D_2 \quad (k = 2, j = 0) &\rightarrow [0(3-1) + 2 - 1]3 = 3 \in F_1 \\
1 \in D_1 + 2 \quad (k = 1, j = 2) &\rightarrow (2-2)3 + 1 = 1 \in F_1 \\
1 \in D_1 + 3 \quad (k = 1, j = 3) &\rightarrow (3-2)3 + 1 = 4 \in F_1 \\
1 \in D_2 + 4 \quad (k = 2, j = 4) &\rightarrow (4-2)3 + 2 = 8 \in F_1
\end{aligned}$$

We use the same method to obtain F_2, F_3, F_4 and F_5. Finally, we get the index array

$1, 5, 6, 11$	$1, 3, 4, 8$	$4, 7, 9, 11$	$2, 3, 7, 10$	$0, 5, 9, 10$	$0, 2, 6, 8$
0	6	1	4	7	10
3	9	2	5	8	11

that represents the generator matrix. ■

2.6 Decoding Algorithm

In this section we introduce a decoding algorithm for the codes obtained in Section 2.3. We study the MDS case, that is when $r = 2$. Then, the code is an $\mathbb{F}_2$-linear code with length n, dimension $n-2$ and minimum distance 3 over $\mathbb{F}_2^b$, with $b = n/2$. The code

$\mathcal{C}_{\mathbb{F}_2^b}$ corrects one single error. However, one error in this case corresponds to b consecutive errors for the linear code $\mathcal{C}_{\mathbb{F}_2}$ over $\mathbb{F}_2$.

Let the index array A_H given in Table 2.5 represent the parity-check matrix of the code. Suppose we receive the word

$$\boldsymbol{x} = \begin{bmatrix} \xi_0 & x_{1,0} & x_{2,0} & \cdots & x_{b-1,0} \\ \xi_1 & x_{1,1} & x_{2,1} & \cdots & x_{b-1,1} \\ \cdots & \cdots & \cdots & & \cdots \\ \xi_{n-1} & x_{1,n-1} & x_{2,n-1} & \cdots & x_{b-1,n-1} \end{bmatrix}.$$

We write the codeword on the index array, corresponding each bit of the codeword with each cell of the array in Table 2.5.

0	ξ_0	1	ξ_1	$\cdots$	$n-1$ $\quad \xi_{n-1}$
D_1	$x_{1,0}$	D_1+1	$x_{1,1}$	$\cdots$	D_1+n-1 $\quad x_{1,n-1}$
D_2	$x_{2,0}$	D_2+1	$x_{2,1}$	$\cdots$	D_2+n-1 $\quad x_{2,n-1}$
$\vdots$		$\vdots$			$\vdots$
D_{b-1}	$x_{b-1,0}$	$D_{b-1}+1$	$x_{b-1,1}$	$\cdots$	$D_{b-1}+n-1$ $\quad x_{b-1,n-1}$

We only consider the cells where the corresponding bit of the codeword is nonzero. Now, we construct the ith component of the vector of syndromes $s = (s_0, s_1, \ldots, s_{n-1})$ in the following way:

$$s_i = \left(\sum_{j=0}^{n-1} \sum_{k=1}^{b-1} x_{k,j} N_{k,j}^i + \xi_i \right) \bmod 2, \quad \text{for} \quad i \in \mathbb{Z}_n, \tag{2.4}$$

where

$$N_{k,j}^i = \begin{cases} 1, & \text{if } i \in D_k + j, \\ 0, & \text{otherwise.} \end{cases}$$

Basically, s_i is the number of times that $i \in \mathbb{Z}_n$ appears in the index array.

If $s_{i_t} \neq 0$, for $t \in \{0, 1, \ldots, l\}$ with $l < n$ we know we have one error related with every index i_t, that is, the index i_t appears too many or not enough times in the index array. We locate these errors in one column of the array, since we can correct only one error in the word. As every column of the index array corresponds to a symbol of the codeword, we know where the codeword has the error. The position of the error in the symbol depends on the position of the cell with errors in the column. Let us see an example to illustrate this idea.

Example 2.9: Consider the index array given in Example 2.1:

0	1	2	3	4	5
$4,5$	$5,0$	$0,1$	$1,2$	$2,3$	$3,4$
$1,3$	$2,4$	$3,5$	$4,0$	$5,1$	$0,2$

This array represents the parity-check matrix of an $\mathbb{F}_2$-linear code of length 6, with dimension 4 and minimum distance 3 over $\mathbb{F}_2^3$.

If we receive the word

$$\boldsymbol{y} = [\ 010 \quad 101 \quad 011 \quad 010 \quad 001 \quad 001\ ]$$

and we know there is one error, we can correct it.

We write the word on the index array and we only consider the cells with a 1.

~~0~~ 0	1 1	~~2~~ 0	~~3~~ 0	~~4~~ 0	~~5~~ 0
$4,5$ 1	~~5~~,~~0~~ 0	$0,1$ 1	$1,2$ 1	~~2~~,~~3~~ 0	~~3~~,~~4~~ 0
~~1~~,~~3~~ 0	$2,4$ 1	$3,5$ 1	~~4~~,~~0~~ 0	$5,1$ 1	$0,2$ 1

Now we count the number of times i appears in the array, for $i \in \mathbb{Z}_6$. If we compute these numbers modulo 2, we obtain the vector of syndromes. On the other hand, we can compute the components of the vector of syndromes $\boldsymbol{s}$ using expression (2.4). The components of the vector of syndromes are given by,

$$s_0 = 2 \bmod 2 = 0,$$
$$s_1 = 4 \bmod 2 = 0,$$
$$s_2 = 3 \bmod 2 = 1,$$
$$s_3 = 1 \bmod 2 = 1,$$
$$s_4 = 2 \bmod 2 = 0,$$
$$s_5 = 3 \bmod 2 = 1.$$

The error is in one column where 2, 3 and 5 appear and the other indices are not affected. Then, the error should be in the column:

2
$0,1$
$3,5$

For example, the error could not be in the column

4
2, 3
5, 1

since the error in 5 would affect to 1, and we obtain no errors in 1.

Then, the error is in the 2nd symbol of the word in the 0th and 2nd position. If we had 1 we change it by 0, and vice versa. Then, the corresponding corrected codeword is

$$\boldsymbol{y} = [\ 010 \quad 101 \quad 110 \quad 010 \quad 001 \quad 001\ ].$$ ■

Chapter 3

MDS Codes Based on Superregular Matrices

3.1 Introduction

In this chapter we introduce an explicit method to construct the parity-check matrix of $\mathbb{F}_q$-linear MDS codes, where $\mathbb{F}_q$ is the Galois field of q elements. Given a superregular matrix over a finite field, we can obtain a superregular block matrix composed by powers of the companion matrix of a primitive polynomial. This superregular block matrix helps us to construct the parity-check matrix of $\mathbb{F}_q$-linear MDS codes [10, 11, 12].

Several constructions of MDS block codes based on superregular matrices have been proposed (see, for instance, [29, 48]). For example, Climent, Napp, Perea and Pinto used superregular matrices to construct 2D convolutional MDS codes [15]. Our aim is to extend this idea using the characterization given in Theorem 1.10 in order to obtain $\mathbb{F}_q$-linear codes which are also MDS.

The next definition given in [47] introduces the concept of superregular matrix.

Definition 3.1: A matrix A is said to be a **superregular matrix** if every square submatrix of A is nonsingular.

We emphasize that some other authors have used the term superregular to define a related but different type of matrices, see for instance [25, 57]. This type of matrices is not suitable to construct $\mathbb{F}_q$-linear MDS codes using Theorem 1.10, as we will see in Example 3.2 below.

Due to Definition 3.1 and Theorem 1.10, we introduce the definition of superregular

block matrix.

Definition 3.2: A matrix $\mathbf{A} \in \mathrm{Mat}_{bt\times bm}(\mathbb{F}_q)$ is said to be a **superregular b-block matrix** if every square submatrix of $\mathbf{A}$ consisting of full blocks submatrices of size $b \times b$ is nonsingular over $\mathbb{F}_q$.

Now, Theorem 1.10, can be expressed in the following way.

Corollary 3.1: *Let $\boldsymbol{H} = [\ \boldsymbol{A} \quad I_{(n-k)b}\ ]$ be an $(n-k)b \times nb$ systematic parity-check matrix of an $\mathbb{F}_q$-linear code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[n,k]$ over $\mathbb{F}_q^b$. Then $\mathcal{C}_{\mathbb{F}_q^b}$ is MDS over $\mathbb{F}_q^b$ if and only if $\boldsymbol{A}$ is a superregular b-block matrix.*

In Example 1.9 the matrix $\mathbf{A}$ given by

$$\mathbf{A} = \left[\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ \hdashline 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right],$$

is a superregular 2-block matrix. Therefore, according to Corollary 3.1, the code whose parity-check matrix is $\mathbf{H} = [\ \mathbf{A} \quad I_4\ ]$ is MDS over $\mathbb{F}_2^2$.

3.2 Minor Constructions

Let $p(\mathsf{x}) = p_0 + p_1\mathsf{x} + \cdots + p_{b-1}\mathsf{x}^{b-1} + \mathsf{x}^b \in \mathbb{F}_q[\mathsf{x}]$. Remember that the companion matrix of $p(\mathsf{x})$ is given by

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -p_{b-2} \\ 0 & 0 & \cdots & 1 & -p_{b-1} \end{bmatrix}.$$

Moreover, if $p(\mathsf{x})$ is a primitive polynomial, it is well known that (see [31])

$$\mathbb{F}_q[C] = \{0, I, C, C^2, \ldots, C^{q^b-2}\} \approx \mathbb{F}_{q^b}. \tag{3.1}$$

The following theorems provide different methods to construct parity-check matrices of $\mathbb{F}_q$-linear MDS codes.

Theorem 3.1: *Let C be the companion matrix of a primitive polynomial of degree b with coefficients in $\mathbb{F}_q$. The matrix $\boldsymbol{H} = [\ \boldsymbol{A} \quad I_b\ ]$ with*

$$\boldsymbol{A} = \begin{bmatrix} C^{i(0,0)} & C^{i(0,1)} & \dots & C^{i(0,m-1)} \end{bmatrix}$$

where $i(0,j) \in \{0,1,2,\dots,q^b-2\}$ with $j \in \{0,1,\dots,m-1\}$, is the parity-check matrix of an $\mathbb{F}_q$-linear MDS code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[m+1, m, 2]$ over $\mathbb{F}_q^b$. Moreover, the dual code of $\mathcal{C}_{\mathbb{F}_q^b}$ is an $\mathbb{F}_q$-linear MDS code with parameters $[m+1, 1, m+1]$ over $\mathbb{F}_q^b$.

Proof: According to expression (3.1), all matrices $C^{i(0,j)}$ with $j = 0,1,2,\dots,m-1$, are nonsingular, then $\mathbf{A}$ is a superregular b-block matrix. Consequently, the theorem follows by Corollary 3.1.

For the dual code of $\mathcal{C}_{\mathbb{F}_q^b}$, the parity-check matrix is

$$\mathbf{G} = [\ I_{mb} \quad -\mathbf{A}^T\ ].$$

Since all the block minors in the matrix $-\mathbf{A}^T$ have the same properties as the ones in matrix $\mathbf{A}$, according to Corollary 3.1 the dual code $\mathcal{C}_{\mathbb{F}_q^b}^{\perp}$ is MDS over $\mathbb{F}_q^b$. □

Theorem 3.2: *Let C be the companion matrix of a primitive polynomial of degree b with coefficients in $\mathbb{F}_q$. For $i(0,r) - i(0,s) \neq i(1,r) - i(1,s) \bmod (q^b-1)$ with $i(0,r), i(0,s), i(1,r), i(1,s) \in \{0,1,2,\dots,q^b-2\}$, the matrix $\boldsymbol{H} = [\ \boldsymbol{A} \quad I_{2b}\ ]$ with*

$$\boldsymbol{A} = \begin{bmatrix} C^{i(0,0)} & C^{i(0,1)} & \dots & C^{i(0,m-1)} \\ C^{i(1,0)} & C^{i(1,1)} & \dots & C^{i(1,m-1)} \end{bmatrix}$$

is the parity-check matrix of an $\mathbb{F}_q$-linear MDS code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[m+2, m, 3]$ over $\mathbb{F}_q^b$. Moreover, the dual code of $\mathcal{C}_{\mathbb{F}_q^b}$ is an $\mathbb{F}_q$-linear MDS code with parameters $[m+2, 2, m+1]$ over $\mathbb{F}_q^b$.

Proof: According to expression (3.1), every 1×1 block minor is a nonsingular matrix.

Every 2×2 block matrix has the form

$$\mathbf{M} = \begin{bmatrix} C^{i(0,r)} & C^{i(0,s)} \\ C^{i(1,r)} & C^{i(1,s)} \end{bmatrix}$$

and by the Schur complement

$$\begin{aligned} \det(\mathbf{M}) &= \det\left(C^{i(0,r)}\right) \cdot \det\left(C^{i(1,s)} - C^{i(1,r)} \cdot C^{-i(0,r)} \cdot C^{i(0,s)}\right) \\ &= \det\left(C^{i(0,r)}\right) \cdot \det\left(C^{i(1,s)} - C^{i(1,r)-i(0,r)+i(0,s)}\right). \end{aligned}$$

Now, by expression (3.1), $C^{i(0,r)}$ is nonsingular and $\det\left(C^{i(1,s)} - C^{i(1,r)-i(0,r)+i(0,s)}\right) = 0$ if and only if

$$i(1,s) = i(1,r) - i(0,r) + i(0,s) \bmod (q^b - 1),$$

which is a contradiction. Therefore, $\det(\mathbf{M}) \neq 0$ and $\mathbf{M}$ is a superregular b-block matrix. Then, by Corollary 3.1 the code $\mathcal{C}_{\mathbb{F}_q^b}$ is an $\mathbb{F}_q$-linear MDS code over $\mathbb{F}_q^b$.

For the dual code the result follows by the same argument used in Theorem 3.1. □

It is possible to find another kind of matrices, not companion matrices, which are useful for this construction. Remember, for example, the EVENODD code [5]. It was an $\mathbb{F}_2$-linear MDS code $\mathcal{C}_{\mathbb{F}_2^{p-1}}$ with parameters $[p+2, p, 3]$ over $\mathbb{F}_2^{p-1}$, with p a prime number. The parity-check matrix was given by

$$\mathbf{H}_{EVODD} = \left[\begin{array}{c:c:c:c:c:c:c} I_{p-1} & I_{p-1} & I_{p-1} & \cdots & I_{p-1} & I_{p-1} & O_{p-1} \\ I_{p-1} & T_{p-1}^{(1)} & T_{p-1}^{(2)} & \cdots & T_{p-1}^{(p-1)} & O_{p-1} & I_{p-1} \end{array}\right],$$

where $T^{(l)} = \left[t_{i,j}^{(l)}\right]$, with

$$t_{i,j}^{(l)} = \begin{cases} 1, & \text{if} \quad j \neq p-l, \quad \text{and} \quad j - i \equiv l \pmod{p}, \\ 1, & \text{if} \quad j = p-l, \\ 0, & \text{otherwise.} \end{cases}$$

The parity-check matrix is in systematic form $\mathbf{H}_{EVODD} = [\ \mathbf{A} \quad I_{2(p-1)}\ ]$, and it can be checked that the matrix $\mathbf{A}$, composed by identity matrices and the $T_{p-1}^{(l)}$ matrices, is a superregular $(p-1)$-block matrix.

3.3 Main Construction

Let C be the companion matrix of a primitive polynomial of degree b in $\mathbb{F}_q[\mathsf{x}]$. Let $\mathbb{F}_{q^b}$ be the Galois field of q^b elements and $\mathbb{F}_q[C] = \{O, I, C, C^2, \ldots, C^{q^b-2}\}$. Due to expression (3.1) we have the following result, whose proof is straightforward.

Theorem 3.3: *Let $\alpha \in \mathbb{F}_{q^b}$ be a fixed primitive element, the following statements hold.*

(a) *The map $\psi : \mathbb{F}_{q^b} \to \mathbb{F}_q[C]$, with $\psi(\alpha) = C$, is a field isomorphism.*

(b) *The map*

$$\begin{array}{rccc} \Psi: & \mathrm{Mat}_{t\times m}(\mathbb{F}_{q^b}) & \longrightarrow & \mathrm{Mat}_{t\times m}(\mathbb{F}_q[C]) \\ & A=[\alpha_{ij}] & \to & \Psi(A)=[\psi(\alpha_{ij})] \end{array} \tag{3.2}$$

is a ring isomorphism.

Given the ring isomorphism in Theorem 3.3 we introduce the next theorem.

Theorem 3.4: *If $A = [\alpha_{ij}] \in \mathrm{Mat}_{t\times m}(\mathbb{F}_{q^b})$ is a superregular matrix over $\mathbb{F}_{q^b}$, then $\boldsymbol{H} = [\ \mathbf{A} \quad I_{tb}\ ]$, where $\mathbf{A} = [\psi(\alpha_{ij})]$, is the parity-check matrix of an $\mathbb{F}_q$-linear MDS code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[m+t, m, t+1]$ over $\mathbb{F}_q^b$. Moreover, the dual code $\mathcal{C}_{\mathbb{F}_q^b}^{\perp}$ is an $\mathbb{F}_q$-linear MDS code with parameters $[m+t, t, m+1]$ over $\mathbb{F}_q^b$.*

Proof: Since $A \in \mathrm{Mat}_{t\times m}(\mathbb{F}_{q^b})$ is a superregular matrix over $\mathbb{F}_{q^b}$, we can say that $\mathbf{A} = \Psi(A) \in \mathrm{Mat}_{t\times m}(\mathbb{F}_q[C])$ is a superregular b-block matrix, due to the properties of the isomorphisms.

Finally, according to Corollary 3.1, the code is MDS. Since the transpose of a superregular matrix is also a superregular matrix, the dual code is MDS as well. □

The following example illustrates this construction.

Example 3.1: Consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x}^2 + \mathsf{x}^3 \in \mathbb{F}_2[\mathsf{x}]$. Then, its companion matrix is given by

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

We consider the superregular matrix over $\mathbb{F}_{2^3}$, $A = \begin{bmatrix} \alpha & 1 \\ 1 & \alpha^2 \end{bmatrix}$, where $\alpha \in \mathbb{F}_{2^3}$ is a fixed primitive element. Due to the ring isomorphism in expression (3.2) we can obtain

$$\Psi(A) = \mathbf{A} = \begin{bmatrix} C & I_3 \\ I_3 & C^2 \end{bmatrix}$$

where

$$\psi(\alpha^l) = C^l \quad \text{with} \quad l \geq 0 \quad \text{and} \quad \psi(0) = O.$$

According to Theorem 3.4, the matrix

$$\mathbf{H} = \left[\begin{array}{cc:c} C & I_3 & I_6 \\ I_3 & C^2 & \end{array}\right] = \left[\begin{array}{ccc:ccc:ccc:ccc} 0&0&1&1&0&0&1&0&0&0&0&0 \\ 1&0&0&0&1&0&0&1&0&0&0&0 \\ 0&1&1&0&0&1&0&0&1&0&0&0 \\ \hdashline 1&0&0&0&1&1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0&0&0&0&1&0 \\ 0&0&1&1&1&1&0&0&0&0&0&1 \end{array}\right]$$

is the parity-check matrix of an $\mathbb{F}_2$-linear MDS code $\mathcal{C}_{\mathbb{F}_2^3}$ with parameters $[4, 2, 3]$ over $\mathbb{F}_2^3$.

Furthermore, the parity-check matrix of $\mathcal{C}_{\mathbb{F}_2^3}^{\perp}$, or equivalently the generator matrix of $\mathcal{C}_{\mathbb{F}_2^3}$, has the form

$$\mathbf{G} = \left[\begin{array}{c:cc} I_6 & C^T & I_3 \\ & I_3 & (C^2)^T \end{array}\right] = \left[\begin{array}{ccc:ccc:ccc:ccc} 1&0&0&0&0&0&0&1&0&1&0&0 \\ 0&1&0&0&0&0&0&0&1&0&1&0 \\ 0&0&1&0&0&0&1&0&1&0&0&1 \\ \hdashline 0&0&0&1&0&0&1&0&0&0&0&1 \\ 0&0&0&0&1&0&0&1&0&1&0&1 \\ 0&0&0&0&0&1&0&0&1&1&1&1 \end{array}\right]$$

so, the dual code $\mathcal{C}_{\mathbb{F}_2^3}^{\perp}$ is also an $\mathbb{F}_2$-linear MDS code with parameters $[4, 2, 3]$ over $\mathbb{F}_2^3$. ■

Some authors use a definition of superregular matrix which is different from the definition given in this work (see, for instance, [25, 57]). As we can see in the following example the concept of superregular matrix in the sense of [25, 57] is not suitable for this construction.

Example 3.2: Consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^3 \in \mathbb{F}_2[\mathsf{x}]$ whose companion matrix is

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The matrix $A = \begin{bmatrix} 1 & 0 \\ \alpha & 1 \end{bmatrix}$ where $\alpha \in \mathbb{F}_{2^3}$ is a fixed primitive element, is a superregular matrix over $\mathbb{F}_{2^3}$ in the sense of [25, 57]. Using the same ring isomorphism introduced in Example 3.1 and according to Theorem 3.4, the parity-check matrix of $\mathcal{C}_{\mathbb{F}_2^3}$ is given by

$$\mathbf{H} = \left[\begin{array}{cc:c} I_3 & O & I_6 \\ C & I_3 & \end{array}\right] = \left[\begin{array}{ccc:ccc:ccc:ccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

It is possible to check that the parameters are $[4, 2, 2]$, then the $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^3}$ is not MDS over $\mathbb{F}_2^3$. ■

The ring isomorphism defined in expression (3.2) comes from a field isomorphism where we use powers of the companion matrix of a primitive polynomial as images of powers of a primitive element. If we use a polynomial which is not primitive, we cannot construct the ring isomorphism, and we cannot assure that Theorem 3.4 holds. That is, the condition that C is a companion matrix of a primitive polynomial is a necessary condition so it ensures that we construct an isomorphism.

Example 3.3: Consider the polynomial $p(\mathsf{x}) = \mathsf{x} + \mathsf{x}^2 + \mathsf{x}^3 \in \mathbb{F}_2[\mathsf{x}]$ whose companion matrix is given by:

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

The matrix

$$A = \begin{bmatrix} \alpha & \alpha \\ 1 & \alpha^3 \end{bmatrix}$$

where $\alpha \in \mathbb{F}_{2^3}$ is a fixed primitive element, is a superregular matrix over $\mathbb{F}_2^3$. If we construct the parity-check matrix, as we did in Example 3.1 we have

$$\mathbf{H} = \left[\begin{array}{ccc:ccc:c:ccc:ccc} 0&0&0&0&0&0&1&0&0&0&0&0\\ 1&0&1&1&0&1&0&1&0&0&0&0\\ 0&1&1&0&1&1&0&0&1&0&0&0\\ \hdashline 1&0&0&0&0&0&0&0&0&1&0&0\\ 0&1&0&1&1&0&0&0&0&0&1&0\\ 0&0&1&1&0&1&0&0&0&0&0&1 \end{array}\right].$$

The corresponding generator matrix is given by

$$\mathbf{G} = \left[\begin{array}{ccc:ccc:c:ccc:ccc} 1&0&0&0&0&0&0&1&0&1&0&0\\ 0&1&0&0&0&0&0&0&1&0&1&0\\ 0&0&1&0&0&0&0&1&1&0&0&1\\ \hdashline 0&0&0&1&0&0&0&1&0&0&1&1\\ 0&0&0&0&1&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&0&1&1&0&0&1 \end{array}\right].$$

The code $\mathcal{C}_{\mathbb{F}_2^3}$ whose generator matrix is $\mathbf{G}$, is an $\mathbb{F}_2$-linear code with parameters $[4, 2, 2]$ over $\mathbb{F}_2^3$, therefore, the code is not MDS. ■

3.4 Families of Superregular Matrices

The use of known families of superregular matrices in order to obtain MDS codes has been studied by several authors (see for example [29, 48]). We can use some of these constructions to extend the results in Section 3.3 for $\mathbb{F}_q$-linear MDS codes.

3.4.1 Cauchy Matrices

Definition 3.3: A **Cauchy matrix** over $\mathbb{F}_{q^b}$ is a $t \times m$ matrix $A = [\alpha_{ij}]$ where $\alpha_{ij} = (x_i - y_j)^{-1}$ whith x_i, $y_j \in \mathbb{F}_{q^b}$ satisfying the following conditions for $i \in \{0, 1, \ldots, t-1\}$ and $j \in \{0, 1, \ldots, m-1\}$:

- $x_i \neq y_j$

- $x_i \neq x_k$ for $k \in \{0, 1, \ldots, t-1\} \setminus \{i\}$
- $y_j \neq y_l$ for $l \in \{0, 1, \ldots, m-1\} \setminus \{j\}$

A Cauchy matrix is a superregular matrix over $\mathbb{F}_{q^b}$ (see [47, 48]).

Consider the sets of elements $\{u_0, u_1, \ldots, u_{t-1}\}$ and $\{v_0, v_1, \ldots, v_{m-1}\}$, satisfying the following conditions for $i \in \{0, 1, \ldots, t-1\}$ and $j \in \{0, 1, \ldots, m-1\}$:

- $u_i, v_j \in \{0, 1, \ldots, q^b - 2\}$
- $u_i \neq v_j$
- $u_i \neq u_k$ for $k \in \{0, 1, \ldots, t-1\} \setminus \{i\}$
- $v_j \neq v_l$ for $l \in \{0, 1, \ldots, m-1\} \setminus \{j\}$

If C is the companion matrix of a primitive polynomial of degree b with coefficients in $\mathbb{F}_q$, according to Theorem 3.4, the matrix $\left[\begin{array}{cc} \mathbf{M} & I_{tb} \end{array}\right]$, where $\mathbf{M} = [M_{ij}]$, with

$$M_{ij} = (C^{u_i} - C^{v_j})^{-1} \quad \text{for} \quad i = 0, 1, \ldots, t-1 \quad \text{and} \quad j = 0, 1, \ldots, m-1,$$

is the parity-check matrix of an $\mathbb{F}_q$-linear MDS code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[m+t, m, t+1]$ over $\mathbb{F}_q^b$. Since the transpose of a Cauchy matrix is a Cauchy matrix as well, the dual code $\mathcal{C}_{\mathbb{F}_q^b}^{\perp}$ is an $\mathbb{F}_q$-linear MDS code with parameters $[m+t, t, m+1]$.

Example 3.4: For $q = 2$ and $b = 3$, consider $(u_1, u_2) = (2, 5)$ and $(v_1, v_2, v_3) = (0, 1, 3)$. Let C be the companion matrix of a primitive polynomial of degree $b = 3$ in $\mathbb{F}_2[\mathsf{x}]$, then the matrix $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{bmatrix} (C^2 - I_3)^{-1} & (C^2 - C)^{-1} & (C^2 - C^3)^{-1} \\ (C^5 - I_3)^{-1} & (C^5 - C)^{-1} & (C^5 - C^3)^{-1} \end{bmatrix}.$$

According to Theorem 3.4, the matrix $\left[\begin{array}{cc} \mathbf{M} & I_6 \end{array}\right]$ is the parity-check matrix of an $\mathbb{F}_2$-linear MDS code $\mathcal{C}_{\mathbb{F}_2^3}$ with parameters $[5, 3, 3]$ over $\mathbb{F}_2^3$. ■

3.4.2 Vandermonde Matrices

In Section 3.4.1 we presented the family of Cauchy matrices which are superregular and we extended this construction for b-block superregular matrices. We could do the

same with Vandermonde matrices, since they are superregular as well. In this section we present another construction of superregular matrices of size $t \times m$, introduced by Lacan and Fimes [30] and based on Vandermonde matrices.

Consider a Vandermonde matrix

$$V = V(\alpha_1, \alpha_2, \ldots, \alpha_t) = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{t-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{t-1} \\ 1 & \alpha_3 & \alpha_3^2 & \cdots & \alpha_3^{t-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_t & \alpha_t^2 & \cdots & \alpha_t^{t-1} \end{bmatrix}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{q^b}$ are pairwise different. Consider also a matrix

$$A = \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_m \\ \gamma_1^2 & \gamma_2^2 & \cdots & \gamma_m^2 \\ \gamma_1^3 & \gamma_2^3 & \cdots & \gamma_m^3 \\ \vdots & \vdots & & \vdots \\ \gamma_1^t & \gamma_2^t & \cdots & \gamma_m^t \end{bmatrix}$$

where $\gamma_1, \gamma_2, \ldots, \gamma_m \in \mathbb{F}_{q^b}$ are pairwise different. Assume also that $\alpha_i \neq \gamma_j$, for all $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, m$. In this case, $M = V^{-1}A$ is a superregular matrix over $\mathbb{F}_{q^b}$ (see [30]). Furthermore, in order to simplify this construction, we can assume that t is a divisor of $q-1$ and take the elements $(\alpha_1, \alpha_2, \ldots, \alpha_t) = (1, \alpha, \alpha^2, \ldots, \alpha^{t-1})$ where $\alpha \in \mathbb{F}_{q^b}$ and $\text{ord}(\alpha) = t$. Then, the corresponding Vandermonde matrix is given by

$$V(\alpha) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{t-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2t-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{t-1} & \alpha^{2t-2} & \cdots & \alpha^{(t-1)^2} \end{bmatrix}.$$

Then $V^{-1} = \frac{1}{n} V(\alpha^{-1})$.

When we work on block matrices, we can extend this construction. Let C be the companion matrix of a primitive polynomial of degree b in $\mathbb{F}_q[\mathrm{x}]$. Consider the Vandermonde

block matrix

$$\mathbf{V} = \mathbf{V}(u_1, u_2, \ldots, u_t) = \begin{bmatrix} I_b & C^{u_1} & (C^{u_1})^2 & \cdots & (C^{u_1})^{t-1} \\ I_b & C^{u_2} & (C^{u_2})^2 & \cdots & (C^{u_2})^{t-1} \\ I_b & C^{u_3} & (C^{u_3})^2 & \cdots & (C^{u_3})^{t-1} \\ \vdots & \vdots & \vdots & & \vdots \\ I_b & C^{u_t} & (C^{u_t})^2 & \cdots & (C^{u_t})^{t-1} \end{bmatrix}$$

where $u_i \in \{0, 1, 2, \ldots, q^b - 2\}$ are pairwise different. Consider also a matrix

$$\mathbf{A} = \begin{bmatrix} C^{v_1} & C^{v_2} & \cdots & C^{v_m} \\ (C^{v_1})^2 & (C^{v_2})^2 & \cdots & (C^{v_m})^2 \\ (C^{v_1})^3 & (C^{v_2})^3 & \cdots & (C^{v_m})^3 \\ \vdots & \vdots & & \vdots \\ (C^{v_1})^t & (C^{v_2})^t & \cdots & (C^{v_m})^t \end{bmatrix}$$

where $v_i \in \{0, 1, 2, \ldots, q^b - 2\}$ are pairwise different. Assume also that $u_i \neq v_j$, for all $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, m$. Thus, $\mathbf{M} = \mathbf{V}^{-1}\mathbf{A}$ is a superregular b-block matrix over $\mathbb{F}_q$.

Therefore, according to Theorem 3.4 the matrix $[\ \mathbf{M} \quad I_{tb}\ ]$ is the parity-check matrix of an $\mathbb{F}_q$-linear MDS code $\mathcal{C}_{\mathbb{F}_q^b}$ with parameters $[m+t, m, t+1]$ over $\mathbb{F}_q^b$ (see next example).

Example 3.5: Let C be the companion matrix of $p(\mathsf{x}) \in \mathbb{F}_2[\mathsf{x}]$ with degree 3. Consider the sets $\{0, 1, 2\}$ and $\{3, 4\}$. We can construct the Vandermonde block matrix

$$\mathbf{V} = \mathbf{V}(0, 1, 2) = \begin{bmatrix} I & I & I \\ I & C & C^2 \\ I & C^2 & C^4 \end{bmatrix}$$

and the block matrix

$$\mathbf{A} = \begin{bmatrix} C^3 & C^4 \\ C^6 & C^8 \\ C^9 & C^{12} \end{bmatrix} = \begin{bmatrix} C^3 & C^4 \\ C^6 & C \\ C^2 & C^5 \end{bmatrix}.$$

Let $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^3 \in \mathbb{F}_2[\mathsf{x}]$, whose companion matrix is

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Then, the matrix $\mathbf{M}$ is given by

$$\mathbf{M} = \mathbf{V}^{-1}\mathbf{A} = \left[\begin{array}{cccccc} 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ \hdashline 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ \hdashline 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right]$$

and is a superregular 3-block matrix over $\mathbb{F}_2$. Therefore, the matrix $\mathbf{H} = [\ \mathbf{M} \quad I_9\ ]$ is the parity-check matrix of an $\mathbb{F}_2$-linear MDS code $\mathcal{C}_{\mathbb{F}_2^3}$ with parameters $[5, 2, 3]$.

3.5 Decoding Algorithm

Blaum, Bruck and Vardy [3] proposed an algorithm for decoding a specific kind of binary codes given the parity-check matrix. We present a similar algorithm for the codes proposed in Section 3.3 in the binary case. That is, $\mathbb{F}_2$-linear codes with parameters $[m+t, m, t+1]$ over $\mathbb{F}_2^b$. For this purpose, instead of working with the parity-check matrix $\mathbf{H} = [\ \mathbf{A} \quad \mathbf{I}\ ]$, where $\mathbf{A}$ is a superregular b-block matrix of size $bt \times bm$, we work with the matrix $H = \Psi^{-1}(\mathbf{H}) = [\ A \quad I\ ]$, with Ψ the ring isomorphism defined in Theorem 3.3. Therefore, the general systematic parity-check matrix H to be considered for the algorithm is given by

$$H = \left[\begin{array}{cccc:cccc} \alpha^{i(0,0)} & \alpha^{i(0,1)} & \cdots & \alpha^{i(0,m-1)} & 1 & 0 & \cdots & 0 \\ \alpha^{i(1,0)} & \alpha^{i(1,1)} & \cdots & \alpha^{i(1,m-1)} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \alpha^{i(t-1,0)} & \alpha^{i(t-1,1)} & \cdots & \alpha^{i(t-1,m-1)} & 0 & 0 & \cdots & 1 \end{array}\right].$$

In this case, a codeword $\boldsymbol{c} = [\ \boldsymbol{c}_0 \quad \boldsymbol{c}_1 \quad \cdots \quad \boldsymbol{c}_{m+t-2} \quad \boldsymbol{c}_{m+t-1}\ ]$, where $\boldsymbol{c}_i \in \mathbb{F}_2^b$, can be expressed as

$$\boldsymbol{c}(\alpha) = [\ c_0(\alpha) \quad c_1(\alpha) \quad \cdots \quad c_{m+t-2}(\alpha) \quad c_{m+t-1}(\alpha)\ ]$$

where $c_i(\alpha) = c_{i,0} + c_{i,1}\alpha + c_{i,2}\alpha^2 + \cdots + c_{i,b-1}\alpha^{b-1}$, for $i \in \{0, 1, \ldots, m+t-1\}$. We can see each symbol as a polynomial modulo $p(\alpha)$, where $p(\mathsf{x})$ is the primitive polynomial used to construct the isomorphism ψ in Theorem 3.3.

Let $\boldsymbol{v}(\alpha) = [\ v_0(\alpha) \quad v_1(\alpha) \quad \cdots \quad v_{m+t-1}(\alpha)\ ]$ be the error-corrupted word. Now we define the syndromes by

$$s_j(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(j,l)} v_l(\alpha) + v_{m+j}(\alpha), \quad j = 0, 1, \ldots, t-1. \tag{3.3}$$

3.5.1 Correcting One Symbol in Error

For the codes constructed in Theorem 3.2, that is, $\mathbb{F}_2$-linear codes with parameters $[m+2, m, 3]$ over $\mathbb{F}_2^b$, with $m \in \mathbb{N}$, we can correct one error.

For this case the parity-check matrix is defined by

$$H = \left[\begin{array}{cccc:cc} \alpha^{i(0,0)} & \alpha^{i(0,1)} & \cdots & \alpha^{i(0,m-1)} & 1 & 0 \\ \alpha^{i(1,0)} & \alpha^{i(1,1)} & \cdots & \alpha^{i(1,m-1)} & 0 & 1 \end{array}\right].$$

Let $\boldsymbol{v}(\alpha) = [\ v_0(\alpha) \quad v_1(\alpha) \quad \cdots \quad v_{m+1}(\alpha)\ ]$ be the error-corrupted word. Now, given the form of the matrix H and expression (3.3) we obtain the syndromes

$$s_0(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(0,l)} v_l(\alpha) + v_m(\alpha),$$
$$s_1(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(1,l)} v_l(\alpha) + v_{m+1}(\alpha).$$

Assuming that the jth column is in error, and that the error value is $e(\alpha)$, we have

$$s_0(\alpha) = \alpha^{i(0,j)} e(\alpha), \tag{3.4}$$

$$s_1(\alpha) = \alpha^{i(1,j)} e(\alpha), \tag{3.5}$$

provided that $0 \leq j \leq m-1$. Combining expressions (3.4) and (3.5) we obtain

$$s_1(\alpha) = \alpha^{i(1,j)-i(0,j)} s_0(\alpha). \tag{3.6}$$

Hence, the location in error is given by the first integer j satisfying expression (3.6). If no such j exists and one of the syndromes is nonzero, there is one error in the corresponding parity column. Otherwise there are more than one error and we cannot correct.

The following example helps us to understand the algorithm shown above.

Example 3.6: We consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^2 \in \mathbb{F}_2[\mathsf{x}]$. The companion matrix of $p(\mathsf{x})$ is given by

$$C = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

If $\alpha \in \mathbb{F}_{2^2}$ is a primitive element, according to Theorem 3.3 the map $\psi : \mathbb{F}_{2^2} \to \mathbb{F}_2[C]$ such that $\psi(\alpha) = C$ can be extended to a ring isomorphism $\Psi : \mathrm{Mat}_{t\times m}(\mathbb{F}_{2^2}) \to \mathrm{Mat}_{t\times m}(\mathbb{F}_2[C])$, such that $\Psi(M) = [\psi(m_{i,j})]$.

Remember that the field $\mathbb{F}_{2^2}$ can be constructed, using the polynomial $p(\mathsf{x})$, in the following way

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

Now, according to Definition 3.1 we know that the matrix

$$A = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}$$

is a superregular matrix over $\mathbb{F}_{2^2}$. Then, the matrix $\mathbf{H} = \Psi([\ A \quad I_2\ ])$ of size 4×8 over $\mathbb{F}_2$ is the parity-check matrix of a $\mathbb{F}_2$-linear MDS code with parameters $[4, 2, 3]$ over $\mathbb{F}_2^2$.

Assume we receive the word

$$\boldsymbol{v} = \left[\begin{array}{c:c:c:c} 10 & 00 & 11 & 11 \end{array}\right]$$

or, in other terms,

$$\boldsymbol{v}(\alpha) = \left[\begin{array}{c:c:c:c} 1 & 0 & 1+\alpha & 1+\alpha \end{array}\right].$$

The parity-check matrix to be considered is

$$\left[\begin{array}{cc:cc} 1 & \alpha & 1 & 0 \\ \alpha & 1 & 0 & 1 \end{array}\right].$$

The components of the vector of syndromes $\boldsymbol{s} = [\ s_0 \quad s_1\ ]$ are

$$s_0(\alpha) = \alpha \quad \text{and} \quad s_1(\alpha) = 1.$$

We check if expression (3.6) holds for any $j = 0, 1$. For $j = 0$ we obtain

$$s_1(\alpha) = 1 \quad \text{and} \quad \alpha^{i(1,0)-i(0,0)} s_0(\alpha) = \alpha^{1-0}\alpha = 1 + \alpha$$

and expression (3.6) does not hold. For $j = 1$ we have

$$s_1(\alpha) = 1 \quad \text{and} \quad \alpha^{i(1,1)-i(0,1)} s_0(\alpha) = \alpha^{0-1}\alpha = 1$$

so expression (3.6) holds. There is one error in position $j = 1$. The error will be

$$e(\alpha) = \alpha^{-i(0,1)} s_0(\alpha) = \alpha^{-1}\alpha = 1$$

and the correct codeword is

$$c(\alpha) = \left[\begin{array}{c:c:c:c} 1 & 1 & 1+\alpha & 1+\alpha \end{array}\right]$$

or

$$\boldsymbol{c} = \left[\begin{array}{c:c:c:c} 10 & 10 & 11 & 11 \end{array}\right]. \qquad \blacksquare$$

3.5.2 Correcting Two Symbols in Error

We consider the $\mathbb{F}_2$-linear codes constructed using Theorem 3.4. We begin with the case where $n = m + 4$, $k = m$ and $d = 5$, for any $m \in \mathbb{N}$. In this case we can correct 2 errors. Let $\boldsymbol{v} = [\, v_0(\alpha) \;\; v_1(\alpha) \;\; \cdots \;\; v_{m+3}(\alpha) \,]$ be the error-corrupted word. Given expression (3.3) we obtain the syndromes

$$s_0(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(0,l)} v_l(\alpha) + v_m(\alpha),$$

$$s_1(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(1,l)} v_l(\alpha) + v_{m+1}(\alpha),$$

$$s_2(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(2,l)} v_l(\alpha) + v_{m+2}(\alpha),$$

$$s_3(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(3,l)} v_l(\alpha) + v_{m+3}(\alpha).$$

Given the vector of syndromes, the next algorithm corrects two errors.

Algorithm 3.1:

(a) If at least two of the syndromes $s_0(\alpha)$, $s_1(\alpha)$, $s_2(\alpha)$, $s_3(\alpha)$ are zero, then there are no errors in the information symbols, and the algorithm stops. Otherwise $l_0 = -1$.

(b) Set $l_0 = l_0 + 1$. If $l_0 = m$ the algorithm stops and we declare there are more than two errors.

(c) Compute the following polynomials

$$\begin{aligned} y_0(\alpha) &= s_0(\alpha) + \alpha^{i(0,l_0)-i(3,l_0)} s_3(\alpha), \\ y_1(\alpha) &= s_1(\alpha) + \alpha^{i(1,l_0)-i(0,l_0)} s_0(\alpha), \\ y_2(\alpha) &= s_2(\alpha) + \alpha^{i(2,l_0)-i(1,l_0)} s_1(\alpha), \\ y_3(\alpha) &= s_3(\alpha) + \alpha^{i(3,l_0)-i(2,l_0)} s_2(\alpha). \end{aligned} \tag{3.7}$$

(d) If at least two of the polynomials $y_j(\alpha)$ and $y_k(\alpha)$, with $k = (j+1) \bmod 4$ in expression (3.7) are zero, there is one single error in the information symbols in position l_0. The error is computed as $e(\alpha) = \alpha^{-i(j,l_0)} s_j(\alpha)$. The algorithm stops. Otherwise, go to the next step.

(e) If $y_2(\alpha) = \alpha^{k_0} y_1(\alpha)$, with

$$\begin{aligned} k_0 &= i(2,l_1) - i(1,l_1) + \mathcal{Z}_\alpha[i(2,l_0) - i(2,l_1) - i(1,l_0) + i(1,l_1)] \\ &\quad - \mathcal{Z}_\alpha[i(1,l_0) - i(1,l_1) - i(0,l_0) + i(0,l_1)] \end{aligned}$$

for some $l_1 \in \{l_0+1, l_0+2, \ldots, m-1\}$ go to the next step. Otherwise go to step (b).

(f) If $y_3(\alpha) = \alpha^{k_1} y_2(\alpha)$, with

$$\begin{aligned} k_1 &= i(3,l_1) - i(2,l_1) + \mathcal{Z}_\alpha[i(3,l_0) - i(3,l_1) - i(2,l_0) + i(2,l_1)] \\ &\quad - \mathcal{Z}_\alpha[i(2,l_0) - i(2,l_1) - i(1,l_0) + i(1,l_1)] \end{aligned}$$

we declare there are errors in positions l_0 and l_1. The algorithm stops. In order to find the errors, we solve the following system:

$$\left.\begin{aligned} s_0(\alpha) &= \alpha^{i(0,l_0)} e_0(\alpha) + \alpha^{i(0,l_1)} e_1(\alpha) \\ s_1(\alpha) &= \alpha^{i(1,l_0)} e_0(\alpha) + \alpha^{i(1,l_1)} e_1(\alpha) \end{aligned}\right\} \tag{3.8}$$

Otherwise go to step (b).

The following theorem shows us that Algorithm 3.1 can correct up to two errors.

Theorem 3.5: *If $\mathcal{C}_{\mathbb{F}_2^b}$ is an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^b$ with parameters $[m+4, m, 5]$, with $m \in \mathbb{N}$ and the parity-check matrix of the code has the form*

$$H = \left[\begin{array}{cccc:cccc} \alpha^{i(0,0)} & \alpha^{i(0,1)} & \cdots & \alpha^{i(0,m-1)} & 1 & 0 & 0 & 0 \\ \alpha^{i(1,0)} & \alpha^{i(1,1)} & \cdots & \alpha^{i(1,m-1)} & 0 & 1 & 0 & 0 \\ \alpha^{i(2,0)} & \alpha^{i(2,1)} & \cdots & \alpha^{i(2,m-1)} & 0 & 0 & 1 & 0 \\ \alpha^{i(3,0)} & \alpha^{i(3,1)} & \cdots & \alpha^{i(3,m-1)} & 0 & 0 & 0 & 1 \end{array}\right],$$

then Algorithm 3.1 corrects up to two errors.

Proof: We check every possible case and we see that in every case, the algorithm corrects the errors.

Case 1: We have one or two errors in the parity symbols. In this case, three or two syndromes are zero, respectively. Then, we would stop in step (a), declaring no errors in the information symbols.

Case 2: We have one single error in the information symbols in the l_0th position. The syndromes are given by

$$\begin{aligned} s_0(\alpha) &= \alpha^{i(0,l_0)} e(\alpha), \\ s_1(\alpha) &= \alpha^{i(1,l_0)} e(\alpha), \\ s_2(\alpha) &= \alpha^{i(2,l_0)} e(\alpha), \\ s_3(\alpha) &= \alpha^{i(3,l_0)} e(\alpha). \end{aligned}$$

It is easy to check that polynomials $y_i(\alpha) = 0$, for $i = 0, 1, 2, 3$, given in step (c) are all zero. Then, the algorithm would run for symbols $0, 1, \ldots, l_0 - 1$ and would stop in step (d), declaring one error in position l_0.

Case 3: We have one error in the information symbols in the l_0th position and one single error in the parity symbols. Without loss of generality we suppose the error in the parity symbol is in the mth position. The syndromes are given by

$$\begin{aligned} s_0(\alpha) &= \alpha^{i(0,l_0)} e_0(\alpha) + e_1(\alpha), \\ s_1(\alpha) &= \alpha^{i(1,l_0)} e_0(\alpha), \\ s_2(\alpha) &= \alpha^{i(2,l_0)} e_0(\alpha), \\ s_3(\alpha) &= \alpha^{i(3,l_0)} e_0(\alpha). \end{aligned}$$

Now, it is possible to check that the polynomials containing $s_0(\alpha)$, that is, $y_0(\alpha)$ and $y_1(\alpha)$, are not zero. The other two polynomials are zero. Then, the algorithm would run for symbols $0, 1, \ldots, l_0 - 1$ and would stop in step (d), declaring one error in position l_0 and another error in a parity symbol.

Case 4: We have two errors in the information symbols in positions l_0 and l_1. The syndromes are given by

$$\begin{aligned} s_0(\alpha) &= \alpha^{i(0,l_0)} e_0(\alpha) + \alpha^{i(0,l_1)} e_1(\alpha), \\ s_1(\alpha) &= \alpha^{i(1,l_0)} e_0(\alpha) + \alpha^{i(1,l_1)} e_1(\alpha), \\ s_2(\alpha) &= \alpha^{i(2,l_0)} e_0(\alpha) + \alpha^{i(2,l_1)} e_1(\alpha), \\ s_3(\alpha) &= \alpha^{i(3,l_0)} e_0(\alpha) + \alpha^{i(3,l_1)} e_1(\alpha). \end{aligned}$$

Then, if we substitute in the polynomials given in step (c), for $y_1(\alpha)$ we obtain

$$\begin{aligned} y_1(\alpha) &= \alpha^{i(1,l_0)} e_0(\alpha) + \alpha^{i(1,l_1)} e_1(\alpha) + \alpha^{i(1,l_0)-i(0,l_0)} [\alpha^{i(0,l_0)} e_0(\alpha) + \alpha^{i(0,l_1)} e_1(\alpha)] \\ &= \alpha^{i(1,l_1)} e_1(\alpha) + \alpha^{i(1,l_0)-i(0,l_0)+i(0,l_1)} e_1(\alpha) \\ &= [\alpha^{i(1,l_1)} + \alpha^{i(1,l_0)-i(0,l_0)+i(0,l_1)}] e_1(\alpha). \end{aligned}$$

Using the properties of the Zech logarithm shown in Definition 2.2 we obtain

$$y_1(\alpha) = \alpha^{i(1,l_1)+\mathcal{Z}_\alpha(i(1,l_0)-i(0,l_0)+i(0,l_1)-i(1,l_1))} e_1(\alpha).$$

We do the same for $y_2(\alpha)$ and obtain the following

$$y_2(\alpha) = \alpha^{i(2,l_1)+\mathcal{Z}_\alpha(i(2,l_0)-i(1,l_0)+i(1,l_1)-i(2,l_1))} e_1(\alpha).$$

As a consequence

$$\begin{aligned} y_2(\alpha) &= \alpha^{i(2,l_1)+\mathcal{Z}_\alpha[i(2,l_0)-i(1,l_0)+i(1,l_1)-i(2,l_1)]-(i(1,l_1)+\mathcal{Z}_\alpha[i(1,l_0)-i(0,l_0)+i(0,l_1)-i(1,l_1)])} y_1(\alpha) \\ &= \alpha^{i(2,l_1)-i(1,l_1)+\mathcal{Z}_\alpha[i(2,l_0)-i(1,l_0)+i(1,l_1)-i(2,l_1)]-\mathcal{Z}_\alpha[i(1,l_0)-i(0,l_0)+i(0,l_1)-i(1,l_1)]} y_1(\alpha). \end{aligned}$$

In the same way, we can obtain

$$y_3(\alpha) = \alpha^{i(3,l_1)-i(2,l_1)+\mathcal{Z}_\alpha[i(3,l_0)-i(2,l_0)+i(2,l_1)-i(3,l_1)]-\mathcal{Z}_\alpha[i(2,l_0)-i(1,l_0)+i(1,l_1)-i(2,l_1)]} y_2(\alpha).$$

These are the expressions given in steps (e) and (f). The algorithm would run for symbols $0, 1, \ldots, l_0 - 1$ and we would have to check for l_0 and for the rest of the information symbols if the expressions in steps (e) and (f) hold. They would hold for l_1. We declare two information errors in positions l_0 and l_1. □

The next example allows us to understand the previous algorithm.

Example 3.7: We consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^4 \in \mathbb{F}_2[\mathsf{x}]$. The companion matrix of $p(\mathsf{x})$ is given by

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

If $\alpha \in \mathbb{F}_{2^4}$ is a primitive element, according to Theorem 3.3 the map $\psi : \mathbb{F}_{2^4} \to \mathbb{F}_2[C]$ such that $\psi(\alpha) = C$ can be extended to a ring isomorphism $\Psi : \mathrm{Mat}_{t\times m}(\mathbb{F}_{2^4}) \to \mathrm{Mat}_{t\times m}(\mathbb{F}_2[C])$, such that $\Psi(M) = [\psi(m_{i,j})]$.

We can consider the following sets $\{0,1,2,3\}, \{4,5,6,7\} \in \mathbb{F}_{2^4}$. According to Section 3.4.1 we can construct the following Cauchy matrix

$$A = \begin{bmatrix} (\alpha^0 - \alpha^4)^{-1} & (\alpha^1 - \alpha^4)^{-1} & (\alpha^2 - \alpha^4)^{-1} & (\alpha^3 - \alpha^4)^{-1} \\ (\alpha^0 - \alpha^5)^{-1} & (\alpha^1 - \alpha^5)^{-1} & (\alpha^2 - \alpha^5)^{-1} & (\alpha^3 - \alpha^5)^{-1} \\ (\alpha^0 - \alpha^6)^{-1} & (\alpha^1 - \alpha^6)^{-1} & (\alpha^2 - \alpha^6)^{-1} & (\alpha^3 - \alpha^6)^{-1} \\ (\alpha^0 - \alpha^7)^{-1} & (\alpha^1 - \alpha^7)^{-1} & (\alpha^2 - \alpha^7)^{-1} & (\alpha^3 - \alpha^7)^{-1} \end{bmatrix}.$$

Now, taking into account that

$$\begin{aligned} \mathbb{F}_{2^4} &= \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\} \\ &= \{0, 1, \alpha, \alpha^2, \alpha^3, 1+\alpha, \alpha+\alpha^2, \alpha^2+\alpha^3, 1+\alpha+\alpha^3, 1+\alpha^2, \alpha+\alpha^3, \\ &\quad 1+\alpha+\alpha^2, \alpha+\alpha^2+\alpha^3, 1+\alpha+\alpha^2+\alpha^3, 1+\alpha^2+\alpha^3, 1+\alpha^3\}, \end{aligned}$$

we obtain that the matrix A is given by

$$A = \begin{bmatrix} \alpha^{14} & 1 & \alpha^5 & \alpha^8 \\ \alpha^5 & \alpha^{13} & \alpha^{14} & \alpha^4 \\ \alpha^2 & \alpha^4 & \alpha^{12} & \alpha^{13} \\ \alpha^6 & \alpha & \alpha^3 & \alpha^{11} \end{bmatrix}.$$

This matrix is a superregular matrix over $\mathbb{F}_{2^4}$. Then, the matrix $\Psi([\ A \quad I_4\ ])$ of size 16×32 is the parity-check matrix of an $\mathbb{F}_2$-linear MDS code with parameters $[8,4,5]$ over $\mathbb{F}_2^4$.

Assume we receive the word

$$\boldsymbol{v}_1 = \left[\begin{array}{c|c|c|c|c|c|c|c} 0001 & 0000 & 0000 & 0000 & 0001 & 0101 & 0011 & 1110 \end{array}\right]$$

or, in other terms,

$$\boldsymbol{v}_1(\alpha) = \left[\begin{array}{c|c|c|c|c|c|c|c} \alpha^3 & 0 & 0 & 0 & \alpha^3 & \alpha+\alpha^3 & \alpha^2+\alpha^3 & 1+\alpha+\alpha^2 \end{array}\right].$$

In order to apply Algorithm 3.1, the parity-check matrix to be considered is

$$H = \left[\begin{array}{cccc|cccc} \alpha^{14} & 1 & \alpha^5 & \alpha^8 & 1 & 0 & 0 & 0 \\ \alpha^5 & \alpha^{13} & \alpha^{14} & \alpha^4 & 0 & 1 & 0 & 0 \\ \alpha^2 & \alpha^4 & \alpha^{12} & \alpha^{13} & 0 & 0 & 1 & 0 \\ \alpha^6 & \alpha & \alpha^3 & \alpha^{11} & 0 & 0 & 0 & 1 \end{array}\right].$$

We calculate the syndromes computing $H\boldsymbol{v}(\alpha)^T$

$$\begin{aligned} s_0(\alpha) &= \alpha^{17} + \alpha^3 = \alpha^2 + \alpha^3, \\ s_1(\alpha) &= \alpha^8 + \alpha + \alpha^3 = 1 + \alpha + \alpha^2 + \alpha^3, \\ s_2(\alpha) &= \alpha^5 + \alpha^2 + \alpha^3 = \alpha + \alpha^3, \\ s_3(\alpha) &= \alpha^9 + 1 + \alpha + \alpha^2 = 1 + \alpha^2 + \alpha^3. \end{aligned}$$

All are different from zero. Then, we start the algorithm with $l_0 = 0$ and we compute the polynomials given in step (c)

$$\begin{aligned} y_0(\alpha) &= s_0(\alpha) + \alpha^{i(0,0)-i(3,0)} s_3(\alpha) = 0, \\ y_1(\alpha) &= s_1(\alpha) + \alpha^{i(1,0)-i(0,0)} s_0(\alpha) = 0, \\ y_2(\alpha) &= s_2(\alpha) + \alpha^{i(2,0)-i(1,0)} s_1(\alpha) = 0, \\ y_3(\alpha) &= s_3(\alpha) + \alpha^{i(3,0)-i(2,0)} s_2(\alpha) = 0. \end{aligned}$$

All of them are zero, that means we have one error in position $l_0 = 0$ as we saw in step (d) of Algorithm 3.1. The error is computed as follows

$$e(\alpha) = \alpha^{-i(0,0)} s_0(\alpha) = \alpha^{-14}(\alpha^2 + \alpha^3) = 1 + \alpha + \alpha^3$$

and then, the correct codeword is

$$\boldsymbol{c}_1(\alpha) = \left[\begin{array}{c|c|c|c|c|c|c|c} 1+\alpha & 0 & 0 & 0 & \alpha^3 & \alpha+\alpha^3 & \alpha^2+\alpha^3 & 1+\alpha+\alpha^2 \end{array}\right],$$

or, in other terms,

$$\boldsymbol{c}_1 = \left[\begin{array}{c|c|c|c|c|c|c|c} 1100 & 0000 & 0000 & 0000 & 0001 & 0101 & 0011 & 1110 \end{array}\right].$$

Assume we receive another word

$$\boldsymbol{v}_2 = \left[\begin{array}{c|c|c|c|c|c|c|c} 0000 & 1000 & 0000 & 0000 & 0001 & 0101 & 0011 & 1110 \end{array}\right]$$

or, in other terms,

$$\boldsymbol{v}_2(\alpha) = \left[\begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 0 & 0 & \alpha^3 & \alpha+\alpha^3 & \alpha^2+\alpha^3 & 1+\alpha+\alpha^2 \end{array}\right].$$

We compute the syndromes

$$\begin{aligned} s_0(\alpha) &= 1+\alpha^3, \\ s_1(\alpha) &= 1+\alpha+\alpha^2, \\ s_2(\alpha) &= 1+\alpha+\alpha^2+\alpha^3, \\ s_3(\alpha) &= 1+\alpha^2. \end{aligned}$$

All are different from zero. Then, we start the algorithm with $l_0 = 0$ and we compute the polynomials given in step (c)

$$\begin{aligned} y_0(\alpha) &= 1+\alpha+\alpha^3, \\ y_1(\alpha) &= 1, \\ y_2(\alpha) &= \alpha^2, \\ y_3(\alpha) &= 1+\alpha+\alpha^2. \end{aligned}$$

None of them is zero, that means we could have one error in $l_0 = 0$, but there should be another error in another position. We have to try whether the conditions in steps (e) and (f) hold for any $l_1 \in \{1,2,3\}$. We start with $l_0 = 0$ and $l_1 = 1$. In this case, we compute the number k_0 given in step (e):

$$\begin{aligned} k_0 &= i(2,1) - i(1,1) + \mathcal{Z}_\alpha\left(i(2,0) - i(2,1) - i(1,0) + i(1,1)\right) \\ &\quad - \mathcal{Z}_\alpha\left(i(1,0) - i(1,1) - i(0,0) + i(0,1)\right) \\ &= 4 + \mathcal{Z}_\alpha(6) - 13 - \mathcal{Z}_\alpha(-22) = 4 + \mathcal{Z}_\alpha(6) - 13 - \mathcal{Z}_\alpha(8) \\ &= 4 + 13 - 13 - 2 = 2 \end{aligned}$$

So, expression $y_2(\alpha) = \alpha^{k_0} y_1(\alpha)$ holds. Now we have to compute the number k_1 given in step (f):

$$\begin{aligned} k_1 &= i(3,1) - i(2,1) + \mathcal{Z}_\alpha\left(i(3,0) - i(3,1) - i(2,0) + i(2,1)\right) \\ &\quad - \mathcal{Z}_\alpha\left(i(2,0) - i(2,1) - i(1,0) + i(1,1)\right) \\ &= 1 + \mathcal{Z}_\alpha(7) - 4 - \mathcal{Z}_\alpha(6) = 1 + 9 - 4 - 13 = -7 \end{aligned}$$

Now, expression $y_3(\alpha) = \alpha^{k_1} y_2(\alpha)$ holds and, therefore, we declare errors in positions $l_0 = 0$ and $l_1 = 1$. In order to find the errors we solve the system in expression (3.8)

$$\left.\begin{aligned} 1 + \alpha^3 &= \alpha^{14} e_0(\alpha) + e_1(\alpha) \\ 1 + \alpha + \alpha^2 &= \alpha^5 e_0(\alpha) + \alpha^{13} e_1(\alpha) \end{aligned}\right\}$$

and we obtain the errors

$$e_0(\alpha) = 1 + \alpha \quad \text{and} \quad e_1(\alpha) = 1.$$

Then, the correct codeword is

$$c = \left[\; 1+\alpha \;\vdots\; 0 \;\vdots\; 0 \;\vdots\; 0 \;\vdots\; \alpha^3 \;\vdots\; \alpha + \alpha^3 \;\vdots\; \alpha^2 + \alpha^3 \;\vdots\; 1 + \alpha + \alpha^2 \;\right]. \qquad \blacksquare$$

3.5.3 Correcting Three Symbols in Error

We consider codes constructed in Theorem 3.4, with $n = m + 6$, $k = m$ and $d = 7$, for $m \in \mathbb{N}$. In this case we can correct 3 errors. Let $\boldsymbol{v}(\alpha) = [\; v_0(\alpha) \;\; v_1(\alpha) \;\; \cdots \;\; v_{m+5}(\alpha) \;]$ be the error-corrupted word. Given the syndromes in expression (3.3) we obtain

$$s_0(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(0,l)} v_l(\alpha) + v_m(\alpha),$$

$$s_1(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(1,l)} v_l(\alpha) + v_{m+1}(\alpha),$$

$$s_2(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(2,l)} v_l(\alpha) + v_{m+2}(\alpha),$$

$$s_3(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(3,l)} v_l(\alpha) + v_{m+3}(\alpha),$$

$$s_4(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(4,l)} v_l(\alpha) + v_{m+4}(\alpha),$$

$$s_5(\alpha) = \sum_{l=0}^{m-1} \alpha^{i(5,l)} v_l(\alpha) + v_{m+5}(\alpha).$$

In this case, the algorithm is quite complicated, so we show the idea of the algorithm using a Vandermonde matrix instead of a general matrix A. Then, the parity-check matrix to be considered is

$$H = \left[\begin{array}{ccccc:c} 1 & \alpha & \alpha^2 & \dots & \alpha^{(m-1)} & \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(m-1)} & \\ 1 & \alpha^3 & \alpha^6 & \dots & \alpha^{3(m-1)} & \\ 1 & \alpha^4 & \alpha^8 & \dots & \alpha^{4(m-1)} & I_6 \\ 1 & \alpha^5 & \alpha^{10} & \dots & \alpha^{5(m-1)} & \\ 1 & \alpha^6 & \alpha^{12} & \dots & \alpha^{6(m-1)} & \end{array}\right]. \tag{3.9}$$

The syndromes are computed as follows

$$s_j(\alpha) = \sum_{l=0}^{m-1} \alpha^{(j+1)l} v_l(\alpha) + v_{m+j}(\alpha) \quad \text{for} \quad i = 0, 1, \dots, 5.$$

The following algorithm shows us how to correct three errors given the vector of syndromes.

Algorithm 3.2:

(a) If at least three of the syndromes $s_0(\alpha)$, $s_1(\alpha)$, $s_2(\alpha)$, $s_3(\alpha)$, $s_4(\alpha)$, $s_5(\alpha)$ are zero, there are no errors in the information symbols, and the algorithm stops. Otherwise $l_0 = -1$.

(b) Set $l_0 = l_0 + 1$. If $l_0 = m$ the algorithm stops and we declare there are more than three errors.

(c) Compute the following polynomials

$$\begin{aligned} y_0(\alpha) &= s_0(\alpha) + \alpha^{-5l_0} s_5(\alpha), \\ y_1(\alpha) &= s_1(\alpha) + \alpha^{l_0} s_0(\alpha), \\ y_2(\alpha) &= s_2(\alpha) + \alpha^{l_0} s_1(\alpha), \\ y_3(\alpha) &= s_3(\alpha) + \alpha^{l_0} s_2(\alpha), \\ y_4(\alpha) &= s_4(\alpha) + \alpha^{l_0} s_3(\alpha), \\ y_5(\alpha) &= s_5(\alpha) + \alpha^{l_0} s_4(\alpha). \end{aligned} \tag{3.10}$$

(d) If at least three consecutive polynomials in expression (3.10) are zero, there is one single error in the information symbols in position l_0. The error in the information symbol is $e(\alpha) = \alpha^{-l_0(k+1)} s_k(\alpha)$, where k is such that

$$y_k(\alpha) = y_{(k+1) \bmod 6}(\alpha) = 0.$$

Otherwise go to the next step.

(e) If four consecutive polynomials in expression (3.10) are not zero (assume they are $y_k(\alpha), y_{k+1}(\alpha), y_{k+2}(\alpha), y_{k+3}$, and the subindices are computed modulo 6) and

$$z(\alpha) = s_{k+3}(\alpha) + \alpha^{2l_0} s_{k+1}(\alpha) = 0,$$

then we declare one error in the information symbol l_0. The error is computed $e(\alpha) = \alpha^{-l_0(k+2)} s_{k+1}(\alpha)$. Otherwise go to the next step.

(f) If there exists an index l_1 such that three consecutive equations,

$$\begin{aligned} y_1(\alpha) &= \alpha^{l_1} y_0(\alpha), \\ y_2(\alpha) &= \alpha^{l_1} y_1(\alpha), \\ y_3(\alpha) &= \alpha^{l_1} y_2(\alpha), \\ y_4(\alpha) &= \alpha^{l_1} y_3(\alpha), \\ y_5(\alpha) &= \alpha^{l_1} y_4(\alpha), \end{aligned}$$

hold, then there are two errors in the information symbols in positions l_0 and l_1. Otherwise go to next step.

(g) If there exist two indices l_1 and l_2 such that the following equations,

$$\begin{aligned} y_3(\alpha) = &\left(\alpha^{2l_1 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{2l_2 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_1(\alpha) \\ &+ \left(\alpha^{2l_1 - l_2 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{2l_2 - l_1 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_2(\alpha), \\ y_4(\alpha) = &\left(\alpha^{3l_1 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{3l_2 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_1(\alpha) \\ &+ \left(\alpha^{3l_1 - l_2 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{3l_2 - l_1 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_2(\alpha), \\ y_5(\alpha) = &\left(\alpha^{4l_1 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{4l_2 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_1(\alpha) \\ &+ \left(\alpha^{4l_1 - l_2 - \mathcal{Z}_\alpha(l_1 - l_2)} + \alpha^{4l_2 - l_1 - \mathcal{Z}_\alpha(l_2 - l_1)}\right) y_2(\alpha), \end{aligned}$$

hold, then there are three errors in the information symbols in positions l_0, l_1 and l_2. We can compute the errors solving the system:

$$\left.\begin{aligned} s_0(\alpha) &= \alpha^{l_0} e_0(\alpha) + \alpha^{l_1} e_1(\alpha) + \alpha^{l_2} e_2(\alpha) \\ s_1(\alpha) &= \alpha^{2l_0} e_0(\alpha) + \alpha^{2l_1} e_1(\alpha) + \alpha^{2l_2} e_2(\alpha) \\ s_2(\alpha) &= \alpha^{3l_0} e_0(\alpha) + \alpha^{3l_1} e_1(\alpha) + \alpha^{3l_2} e_2(\alpha) \end{aligned}\right\}$$

Otherwise go to step (b).

The next theorem states that Algorithm 3.2 corrects up to three errors.

Theorem 3.6: *If $\mathcal{C}_{\mathbb{F}_2^b}$ is an $\mathbb{F}_2$-linear code over $\mathbb{F}_2^b$ with parameters $[m+6, m, 7]$, with $m \in \mathbb{N}$ and the parity-check matrix of the code has the form given in expression (3.9), then Algorithm 3.2 corrects up to three errors.*

The proof of the theorem involves considering the various cases of errors in the information part and the parity part. It is similar to the proof of Theorem 3.5 and it is omitted.

It is possible to extend this idea for decoding codes with higher length. However, the decoding of such schemes grows exponentially with the length.

Chapter 4

LFSR: A Systems Theory Point of View

4.1 Introduction

The main problem considered in symmetric cryptography is to generate sequences $\boldsymbol{r} = (r_0, r_1, r_2, \ldots)$ as random as possible using a short key. There are many ways to solve this complicated task. One approach is through some nonlinear recurrence relation

$$r_{t+l} = f(r_{t+l-1}, r_{t+l-2}, \ldots, r_t), \quad t = 0, 1, \ldots$$

having the initial state $(r_0, r_1, \ldots, r_{l-1})$ as the key. Remember that if the recurrence relation is linear, we say the keystream generator is a **linear feedback shift register** (LFSR) [18, 19, 58], or LFSR for short (see Section 1.4).

The sequences generated by LFSRs have properties desirable for keystreams. However, with the Berlekamp-Massey algoritm [35] it is very easy to compute the feedback polynomial of the LFSR given at least $2\mathcal{L}(\boldsymbol{s})$ bits of the sequence $\boldsymbol{s}$, where $\mathcal{L}(\boldsymbol{s})$ is the linear complexity of the sequence $\boldsymbol{s}$. As a consequence, we could recover the key.

We need to destroy the linearity before the sequence can be used as keystream. One classical approach is to combine several binary LFSRs via a nonlinear Boolean function f (see Figure 1.4). The properties of the output sequence will be good depending on the choice of the function f.

In this chapter we study LFSRs from a different point of view. First we analyze LFSRs as codes over extension alphabets, then the problem of recovering the initial state turns into a decoding problem.

After that, we introduce the concept of generalized LFSRs and we propose a correlation

problem involving generalized LFSRs. Finally, we propose a method to obtain primitive polynomials from other primitive polynomials of smaller degree, and we indicate a way to construct polynomial matrices given its determinant.

4.2 LFSR as an $\mathbb{F}_2$-Linear Code

In this section we consider LFSRs as codes and we present a decoding algorithm for these codes. From now on, we denote the companion matrix by A, due to the conventional notation for systems theory and the matrix C will be the matrix composed by the first column of the identity matrix.

4.2.1 Construction

Let $\mathbb{F}_2$ be the Galois field of two elements. We consider the binary case since it is the most usual in cryptography. Assume that

$$f(\mathsf{x}) = c_0 + c_1\mathsf{x} + c_2\mathsf{x}^2 + \cdots + c_{l-1}\mathsf{x}^{l-1} + \mathsf{x}^l \in \mathbb{F}_2[\mathsf{x}]$$

is the feedback polynomial of an LFSR of length l. Let us consider the companion matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & c_0 \\ 1 & 0 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & 0 & \cdots & 0 & c_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_{l-2} \\ 0 & 0 & 0 & \cdots & 1 & c_{l-1} \end{bmatrix} \in \mathrm{Mat}_{l\times l}(\mathbb{F}_2)$$

of $f(\mathsf{x})$ and the column matrix

$$C = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T \in \mathrm{Mat}_{l\times 1}(\mathbb{F}_2).$$

A way to describe the LFSR is by means of the autonomous system

$$\left.\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t A \\ y_t &= \boldsymbol{x}_t C \end{aligned}\right\} \qquad \text{for} \quad t = 0, 1, 2 \ldots, \tag{4.1}$$

where $\boldsymbol{x}_0$ is the **initial state** of the system and A is called the **transition matrix**. If $\boldsymbol{x}_0 = [\ y_0 \quad y_1 \quad \ldots \quad y_{l-1}\ ]$ is the initial state, the tth stream bit y_t, for $t = l, l+1, l+2, \ldots$

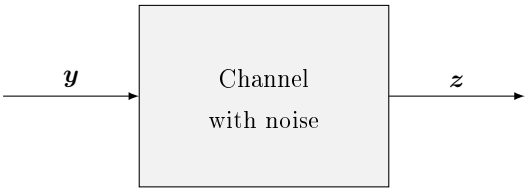


Figure 4.1: *Codeword transmitted over a channel.*

can be computed using the system in expression (4.1). Moreover, if $f(\mathsf{x})$ is a primitive polynomial, then the sequence

$$y_0, y_1, \ldots, y_{l-1}, y_l, \ldots,$$

has maximum period $2^l - 1$; that is

$$y_{2^l-1+t} = y_t, \quad \text{for} \quad t = 0, 1, 2 \ldots$$

Let N be a positive integer such that $l < N < 2^l - 1$. We can compute the output sequence

$$\boldsymbol{y} = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{l-1} & y_l & y_{l+1} & \cdots & y_{N-1} \end{bmatrix},$$

as

$$\boldsymbol{y} = \boldsymbol{x}_0 G,$$

where the matrix G is given by

$$G = \begin{bmatrix} C & AC & A^2C & A^3C & \cdots & A^{N-2}C & A^{N-1}C \end{bmatrix}.$$

This matrix is said to be the **observability matrix** of the system given in expression (4.1).

Assume that we do not know the sequence $\boldsymbol{y}$ nor the initial state $\boldsymbol{x}_0$. Assume also that we know the sequence

$$\boldsymbol{z} = \begin{bmatrix} z_0 & z_1 & z_2 & \cdots & z_{l-1} & z_l & z_{l+1} & \cdots & z_{N-1} \end{bmatrix}$$

which is correlated to the sequence $\boldsymbol{y}$ with correlation probability p (usually $0.5 \leq p \leq 0.75$).

Correlation attacks are often viewed as a decoding problem. Thus, the LFSR sequence $\boldsymbol{y}$ is interpreted as a codeword in the linear code $\mathcal{C}_{\mathbb{F}_2}$ with parameters $[N, l]$ over $\mathbb{F}_2$ generated by the matrix G and the keystream sequence $\boldsymbol{z}$ as the received channel output (see Figure 4.1).

The problem of the attacker can now be formulated as: *Given a received word* $\boldsymbol{z}$*, find the transmitted codeword* $\boldsymbol{y}$. Remember that $\boldsymbol{x}_0$ is the vector whose components are the first l components of $\boldsymbol{y}$.

Assume now that $N = ml$ for some positive integer m. Then

$$\left[\begin{array}{ccccccc} C & AC & A^2C & A^3C & \cdots & A^{l-1}C \end{array}\right] = I_l$$

where I_l denotes the $l \times l$ identity matrix. Consequently

$$\begin{aligned}\left[\begin{array}{ccccccc} A^lC & A^{l+1}C & A^{l+2}C & A^{l+3}C & \cdots & A^{2l-1}C \end{array}\right] \\ = A^l \left[\begin{array}{ccccccc} C & AC & A^2C & A^3C & \cdots & A^{l-1}C \end{array}\right] = A^l\end{aligned}$$

and so on. So, we can write the generator matrix G of $\mathcal{C}_{\mathbb{F}_2}$ as

$$G = \left[\begin{array}{c:cccccc} I_l & A^l & A^{2l} & A^{3l} & \cdots & A^{(m-2)l} & A^{(m-1)l} \end{array}\right].$$

Since the generator matrix is in systematic form, the corresponding parity-check matrix is

$$H = \left[\begin{array}{c:c} \begin{array}{c} \left(A^l\right)^T \\ \left(A^{2l}\right)^T \\ \left(A^{3l}\right)^T \\ \vdots \\ \left(A^{(m-2)l}\right)^T \\ \left(A^{(m-1)l}\right)^T \end{array} & I_{(m-1)l} \end{array}\right].$$

As we saw in Theorem 1.6 if we consider G as the generator matrix of a code $\mathcal{C}_{\mathbb{F}_2}$ over $\mathbb{F}_2$ this code can not be MDS, since it is a binary linear non trivial code. Then, we consider the code over $\mathbb{F}_2^l$, that is, the matrix G is the generator matrix of an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^l}$ with parameters $[m, 1]$ over $\mathbb{F}_2^l$. Due to the fact that G is composed by powers of the companion matrix A and Theorem 3.1, we know the code $\mathcal{C}_{\mathbb{F}_2^l}$ is MDS over $\mathbb{F}_2^l$. Then, the parameters of the code are $[m, 1, m]$ over $\mathbb{F}_2^l$. Due to expression (1.2) we can correct $\left\lfloor \frac{m-1}{2} \right\rfloor$ errors.

4.2.2 Correcting One Symbol in Error

Suppose we have an $\mathbb{F}_2$-linear code with parameters $[3, 1, 3]$ over $\mathbb{F}_2^l$ whose parity-check matrix is

$$H = \left[\begin{array}{c:cc} (A^l)^T & I_l & O \\ (A^{2l})^T & O & I_l \end{array}\right]. \tag{4.2}$$

Since the minimum distance is 3, we can only correct one error.

Suppose we receive the word $\boldsymbol{z} = [\ \boldsymbol{z}_0 \quad \boldsymbol{z}_1 \quad \boldsymbol{z}_2\ ]$, then the vector of syndromes $\boldsymbol{s} = [\ \boldsymbol{s}_0 \quad \boldsymbol{s}_1\ ]$ is given by

$$\begin{aligned} \boldsymbol{s}_0^T &= (A^l)^T \boldsymbol{z}_0^T + \boldsymbol{z}_1^T, \\ \boldsymbol{s}_1^T &= (A^{2l})^T \boldsymbol{z}_0^T + \boldsymbol{z}_2^T. \end{aligned}$$

We know that $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, where $\boldsymbol{y}$ is the correct codeword and $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2\ ]$ is the error. Then

$$\begin{aligned} \boldsymbol{s}_0^T &= (A^l)^T \boldsymbol{e}_0^T + \boldsymbol{e}_1^T, \\ \boldsymbol{s}_1^T &= (A^{2l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_2^T. \end{aligned} \tag{4.3}$$

The next algorithm, based on the algorithms given in Section 3.5, shows how to correct one symbol in error.

Algorithm 4.1: Assume we know the vector of syndromes $\boldsymbol{s} = [\ \boldsymbol{s}_0 \quad \boldsymbol{s}_1\ ]$. Following the steps below, we can correct one symbol in error for an $\mathbb{F}_2$-linear code whose parity-check matrix has the form given in expression (4.2).

(a) If $\boldsymbol{s}_0 = \boldsymbol{s}_1 = \boldsymbol{0}$, there is no error and $\boldsymbol{z}$ is a codeword. Go to step (d).

(b) If $\boldsymbol{s}_0 \neq \boldsymbol{0}$ or $\boldsymbol{s}_1 \neq \boldsymbol{0}$ we have one error in a parity symbol, and the information symbol $\boldsymbol{z}_0$ is correct. Go to step (d).

(c) If $\boldsymbol{s}_1^T = (A^l)^T \boldsymbol{s}_0^T$, there is one error in the information symbol. The error is given by $\boldsymbol{e}_0^T = (A^{-l})^T \boldsymbol{s}_0^T$. Go to step (d).

(d) End.

The next theorem shows that Algorithm 4.1 corrects up to one error.

Theorem 4.1: *For an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^l}$ with parameters $[3, 1, 3]$ over $\mathbb{F}_2^l$ whose parity-check matrix is the one in expression (4.2), Algorithm 4.1 corrects up to one error.*

PROOF: Assume we receive the word $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, where $\boldsymbol{e} = [\ e_0 \quad e_1 \quad e_2\ ]$. The vector of syndromes $\boldsymbol{s} = [\ \boldsymbol{s}_0 \quad \boldsymbol{s}_1\ ]$ is given in expression (4.3).

If the error is in the first parity symbol, that is $\boldsymbol{e} = [\ \boldsymbol{0} \quad \boldsymbol{e}_1 \quad \boldsymbol{0}\ ]$, the vector of syn-

dromes is given by

$$\boldsymbol{s}_0^T = \boldsymbol{e}_1^T \quad \text{and} \quad \boldsymbol{s}_1^T = \mathbf{0}.$$

If the error is in the second parity symbol, that is $\boldsymbol{e} = [\ \mathbf{0} \quad \mathbf{0} \quad \boldsymbol{e}_2\ ]$, the vector syndromes is given by

$$\boldsymbol{s}_0^T = \mathbf{0} \quad \text{and} \quad \boldsymbol{s}_1^T = \boldsymbol{e}_2^T.$$

If one of the syndromes is zero, we have one error in a parity symbol, as step (b) in Algorithm 4.1 establishes.

If the error is in the information symbol, that is $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \mathbf{0} \quad \mathbf{0}\ ]$, the vector of syndromes is given by

$$\boldsymbol{s}_0^T = (A^l)^T \boldsymbol{e}_0^T \quad \text{and} \quad \boldsymbol{s}_1^T = (A^{2l})^T \boldsymbol{e}_0^T.$$

Combining both expressions, we have $\boldsymbol{s}_1^T = (A^l)^T \boldsymbol{s}_0^T$, which is the expression that appears in step (c) in Algorithm 4.1.

If there is more than one error, the code can not correct, since the minimum distance of the code is 3. □

It is worth noticing that in several cases the algorithm could decode to the wrong codeword. We analyze these cases.

- If we have two errors in the parity symbols, that is $\boldsymbol{e} = [\ \mathbf{0} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2\ ]$, then the components of the vector of syndromes can be expressed as follows

  $$\boldsymbol{s}_0^T = \boldsymbol{e}_1^T \quad \text{and} \quad \boldsymbol{s}_1^T = \boldsymbol{e}_2^T.$$

  If $\boldsymbol{e}_2^T = (A^l)^T \boldsymbol{e}_1^T$, then we correct as if we had one error in the information symbol (see step (c) in Algorithm 4.1).

- If we have two errors; one in the information symbol and one error in the first parity symbol, that is $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \boldsymbol{e}_1 \quad \mathbf{0}\ ]$, then the components of the vector of syndromes are

  $$\boldsymbol{s}_0^T = (A^l)^T \boldsymbol{e}_0^T + \boldsymbol{e}_1^T \quad \text{and} \quad \boldsymbol{s}_1^T = (A^{2l})^T \boldsymbol{e}_0^T.$$

  If $\boldsymbol{e}_1^T = (A^l)^T \boldsymbol{e}_0^T$, then $\boldsymbol{s}_0 = \mathbf{0}$ and we correct as if we had one error in the second parity symbol (see step (b) in Algorithm 4.1).

- If we have two errors; one in the information symbol and one error in the second parity symbol, that is $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \mathbf{0} \quad \boldsymbol{e}_2\ ]$, then the components of the vector of syndromes are

  $$\boldsymbol{s}_0^T = (A^l)^T \boldsymbol{e}_0^T \quad \text{and} \quad \boldsymbol{s}_1^T = (A^{2l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_2^T.$$

	degree			
	7	8	10	12
Random errors	$2\% - 2.5\%$	$1\% - 1.12\%$	$0.3\% - 0.5\%$	$0.07\% - 0.08\%$
Errors with 25% of 1s	$3\% - 3.5\%$	$0.75\% - 0.8\%$	$0.5\% - 0.6\%$	$0.05\% - 0.075\%$

Table 4.1: *Percentage of error.*

If $\boldsymbol{e}_2^T = (A^{2l})^T \boldsymbol{e}_0^T$, then $\boldsymbol{s}_1 = \mathbf{0}$ and we correct as if we had one error in the first parity symbol (see step (b) in Algorithm 4.1).

- If we have three errors, that is $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2\ ]$, then the vector of syndromes is

$$\boldsymbol{s}_0^T = (A^l)^T \boldsymbol{e}_0^T + \boldsymbol{e}_1^T \quad \text{and} \quad \boldsymbol{s}_1^T = (A^{2l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_2^T.$$

If $\boldsymbol{e}_1^T = (A^l)^T \boldsymbol{e}_0^T$ or $\boldsymbol{e}_2^T = (A^{2l})^T \boldsymbol{e}_0^T$, then $\boldsymbol{s}_0 = \mathbf{0}$ or $\boldsymbol{s}_1 = \mathbf{0}$, respectively. If both are zero, we do not correct since we think we have a codeword (see step (a) in Algorithm 4.1). If just one of them is zero, we correct as if we had one error in the parity symbol (see step (b) in Algorithm 4.1). On the other hand, if $\boldsymbol{e}_2^T = (A^l)^T \boldsymbol{e}_1^T$, then $\boldsymbol{s}_1^T = (A^l)^T \boldsymbol{s}_0^T$, we correct as if we had one error in the information symbol (see step (c) in Algorithm 4.1).

At this point we study how often we could find these problems for random errors.

Example 4.1: We consider primitive polynomials with coefficients in $\mathbb{F}_2$ and degrees $7, 8, 10$ and 12. For example, for polynomials of degree 7 we compute the companion matrix A and the corresponding parity-check matrix H in expression (4.2). We generate 100000 random vectors $\boldsymbol{e} \in \mathbb{F}_2^{21}$ several times. As we can see in Table 4.1, for polynomials of degree 7, Algorithm 4.1 decodes to the wrong codeword between 2% and 2.5% of the cases. We usually work with a channel with low probability of flipping the bit, for example $p = 0.25$. Then we also consider when the error has only 25% of 1s. For 100000 random errors of length 21, the average is a little higher, between 3% and 3.5%.

We can check in Table 4.1 that the higher the degree is the smaller is the percentage of error using Algorithm 4.1. ■

As we observe in the previous example, the higher the degree l of the polynomial is, the lower is the probability of receiving a word that is corrected incorrectly.

4.2.3 Correcting Two Symbols in Error

We consider the smallest case for correcting two errors, that is, we have an $\mathbb{F}_2$-linear code with parameters $[5, 1, 5]$ over $\mathbb{F}_2^l$ whose parity-check matrix is

$$H = \left[\begin{array}{c|cccc} (A^l)^T & I_l & O & 0 & 0 \\ (A^{2l})^T & O & I_l & 0 & 0 \\ (A^{3l})^T & O & 0 & I_l & 0 \\ (A^{4l})^T & O & 0 & 0 & I_l \end{array}\right]. \tag{4.4}$$

Suppose we receive the word $\boldsymbol{z} = [\ \boldsymbol{z}_0 \quad \boldsymbol{z}_1 \quad \boldsymbol{z}_2 \quad \boldsymbol{z}_3 \quad \boldsymbol{z}_4\ ]$, then the vector of syndromes $\boldsymbol{s} = [\ \boldsymbol{s}_0 \quad \boldsymbol{s}_1 \quad \boldsymbol{s}_2 \quad \boldsymbol{s}_3\ ]$ is given by

$$\begin{aligned} \boldsymbol{s}_0^T &= (A^l)^T \boldsymbol{z}_0^T + \boldsymbol{z}_1^T, \\ \boldsymbol{s}_1^T &= (A^{2l})^T \boldsymbol{z}_0^T + \boldsymbol{z}_2^T, \\ \boldsymbol{s}_2^T &= (A^{3l})^T \boldsymbol{z}_0^T + \boldsymbol{z}_3^T, \\ \boldsymbol{s}_3^T &= (A^{4l})^T \boldsymbol{z}_0^T + \boldsymbol{z}_4^T. \end{aligned}$$

We know that $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, where $\boldsymbol{y}$ is the correct codeword and $\boldsymbol{e} = [\ \boldsymbol{e}_0 \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2 \quad \boldsymbol{e}_3 \quad \boldsymbol{e}_4\ ]$ is the error. Then

$$\begin{aligned} \boldsymbol{s}_0^T &= (A^l)^T \boldsymbol{e}_0^T + \boldsymbol{e}_1^T, \\ \boldsymbol{s}_1^T &= (A^{2l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_2^T, \\ \boldsymbol{s}_2^T &= (A^{3l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_3^T, \\ \boldsymbol{s}_3^T &= (A^{4l})^T \boldsymbol{e}_0^T + \boldsymbol{e}_4^T. \end{aligned} \tag{4.5}$$

As we did in Section 4.2.2, we propose an algorithm in order to correct two errors.

Algorithm 4.2: Assume we know the vector of syndromes $\boldsymbol{s} = [\ \boldsymbol{s}_0 \quad \boldsymbol{s}_1 \quad \boldsymbol{s}_2 \quad \boldsymbol{s}_3\ ]$. If we follow the steps below, we can correct two symbols in error for an $\mathbb{F}_2$-linear code whose parity-check matrix has the form given in expression (4.4).

(a) If $\boldsymbol{s}_0 = \boldsymbol{s}_1 = \boldsymbol{s}_2 = \boldsymbol{s}_3 = \boldsymbol{0}$, there is no error and $\boldsymbol{z}$ is a codeword. Go to step (e).

(b) If one or two syndromes are not zero we have one or two errors in the parity symbols, therefore, the information symbol $\boldsymbol{z}_0$ is correct. Go to step (e).

(c) If the following equations

$$\begin{aligned} s_1^T &= (A^l)^T s_0^T, \\ s_2^T &= (A^l)^T s_1^T, \\ s_3^T &= (A^l)^T s_2^T, \\ s_0^T &= (A^{-3l})^T s_3^T, \end{aligned} \tag{4.6}$$

hold, there is one error in the information symbol. The error is given by $\boldsymbol{e}_0^T = (A^{-l})^T \boldsymbol{s}_0^T$. Go to step (e).

(d) If two consecutive equations in expression (4.6) hold, and the others do not hold, we have one error in the information symbol and one error in a parity symbol. We can compute the error using the syndrome involved in the equations which hold. Go to step (e).

(e) End.

The next theorem shows that Algorithm 4.2 corrects up to two symbols in error.

Theorem 4.2: *For an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^l}$ with parameters $[5, 1, 5]$ over $\mathbb{F}_2^l$ whose parity-check matrix is the one in expression (4.4), then Algorithm 4.2 corrects up to two errors.*

The proof follows the same idea we used for proving Theorem 4.1. We analyze the possible cases and we check that the algorithm corrects up to two errors. There are cases where Algorithm 4.2 decodes to the wrong codeword, but as it happened for Algorithm 4.1, the probability is very small.

The idea used for Algorithms 4.1 and 4.2 can be extended to correct codes with greater length.

4.3 Generalized LFSR

In this section we introduce a new sort of LFSR that we call **generalized LFSR**, GLFSR for short. The main diference, is that in each round we obtain several symbols instead of one.

4.3.1 Construction

We consider the generalization of the LFSRs in autonomous systems.

Consider the polynomial matrix $P(\mathsf{x}) = \sum_{i=0}^{\delta} P_i \mathsf{x}^i \in \mathrm{Mat}_{n\times n}(\mathbb{F}_2[\mathsf{x}])$, with $P_\delta = I_n$. In this case, the autonomous system representing the GLFSR is

$$\left.\begin{array}{rcl} \boldsymbol{x}_{t+1} & = & \boldsymbol{x}_t A \\ \boldsymbol{y}_t & = & \boldsymbol{x}_t C \end{array}\right\} \quad \text{for} \quad t = 0, 1, 2, \ldots$$

The output sequence $\boldsymbol{y} = [\ y_0 \quad y_1 \quad \cdots \quad y_{N-1}\ ]$ can be computed as,

$$\boldsymbol{y} = \boldsymbol{x}_0 G$$

where $\boldsymbol{x}_0$ is the initial state and

$$G = \left[\ C \quad AC \quad A^2C \quad \cdots \quad A^{N-1}C\ \right]. \tag{4.7}$$

Here the matrices A and C are a generalization of the matrices given in Subsection 4.2.1. They are constructed in the following way,

$$A = \left[\begin{array}{cccc:c} O & O & \cdots & O & P_0 \\ I_n & O & \cdots & O & P_1 \\ O & I_n & \cdots & O & P_2 \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \cdots & O & P_{\delta-2} \\ O & O & \cdots & I_n & P_{\delta-1} \end{array}\right] \quad \text{and} \quad C = \left[\begin{array}{c} I_n \\ O \\ \vdots \\ O \end{array}\right]. \tag{4.8}$$

Let $p(\mathsf{x}) = \det P(\mathsf{x})$. It is evident that the degree of $p(\mathsf{x})$ is δn. By [20] we can deduce also that $p(\mathsf{x}) = \det(\mathsf{x}I - A)$, that is, it is the characteristic polynomial of the matrix A. This polynomial behaves like the feedback polynomial in the LFSR case. We want to hide very large messages, so we need the period of A to be as large as possible.

The next example illustrates the construction of A and C from a given polynomial matrix.

Example 4.2: Consider the polynomial matrix

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x}^2+1 & \mathsf{x} \\ 1 & \mathsf{x}^2+1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathsf{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathsf{x}^2$$

whose determinant is $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^4$. In this example, $\delta = 2$ and $n = 2$. The matrices

A and C can be computed as above

$$A = \left[\begin{array}{c:c} O & P_0 \\ \hdashline I_2 & P_1 \end{array}\right] = \left[\begin{array}{cc:cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ \hdashline 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad C = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline 0 & 0 \\ 0 & 0 \end{array}\right].$$

Notice that in this case the output sequence is a vector whose entries are also vectors, that is, in each round we obtain two symbols instead of one. Let $\boldsymbol{x}_0 = [\ 0 \quad 0 \quad 0 \quad 1\ ]$ be the initial state, then the output sequence is

$$\boldsymbol{y} = \left[\begin{array}{c:c:c:c:c:c:c:c:c:c:c} 00 & 01 & 00 & 11 & 01 & 01 & 11 & 10 & 00 & 10 & \cdots \end{array}\right].$$

■

Theorem 4.3: *Let $P(\mathsf{x}) = \sum_{i=0}^{\delta} P_i \mathsf{x}^i$, $P_\delta = I_n$, be an $n \times n$ polynomial matrix over $\mathbb{F}_2$. If $p(\mathsf{x}) = \det P(\mathsf{x})$ is a primitive polynomial then the matrix A computed as in expression (4.8) has maximal period $2^{\delta n} - 1$.*

The next example shows the different matrices A that we can obtain from the same polynomial.

Example 4.3: Consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^6 \in \mathbb{F}_2[\mathsf{x}]$. We can construct different matrices A according to expression (4.8).

(a) For instance, we can consider the matrix $P(\mathsf{x})$ of size 2×2, whose determinant is the polynomial $p(\mathsf{x})$,

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x}^3 & \mathsf{x}+1 \\ 1 & \mathsf{x}^3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathsf{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \mathsf{x}^2 + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathsf{x}^3.$$

Therefore, the matrices A and C are given by

$$A = \left[\begin{array}{cc:cc:cc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad C = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline 0 & 0 \\ 0 & 0 \\ \hdashline 0 & 0 \\ 0 & 0 \end{array}\right].$$

(b) Let us consider a different polynomial matrix $P(\mathsf{x})$ of size 3×3, whose determinant is also $p(\mathsf{x})$,

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x}^2 & 0 & 1 \\ 1 & \mathsf{x}^2 & 0 \\ 0 & \mathsf{x}+1 & \mathsf{x}^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathsf{x} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathsf{x}^2.$$

In this case, A and C have the form

$$A = \left[\begin{array}{ccc:ccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \quad \text{and} \quad C = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hdashline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right].$$

(c) The determinant of the next polynomial matrix $P(\mathsf{x})$ of size 6×6 is also $p(\mathsf{x})$,

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x} & 0 & 0 & 0 & 1 & 1 \\ 1 & \mathsf{x} & 0 & 0 & 0 & 0 \\ 0 & 1 & \mathsf{x} & 0 & 0 & 0 \\ 0 & 0 & 1 & \mathsf{x} & 0 & 0 \\ 0 & 0 & 0 & 1 & \mathsf{x} & 0 \\ 0 & 0 & 0 & 0 & 1 & \mathsf{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathsf{x}.$$

Now, the matrices A and C are

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad \blacksquare$$

As we can see in the previous example, the matrix A has the same size in each case, but the matrix C does not. In the last case the matrix C has the maximum size, that is, the

same size as A. Then, the sequence obtained is longer than the sequences obtained in the other cases. This happens because the matrix $P(\mathsf{x})$ has the maximum possible size for a matrix with determinant $p(\mathsf{x})$, that is size 6×6. In general, the matrix G in expression (4.7) has size $\delta n \times Nn$. We want our sequences to be as long as possible so we want $p(\mathsf{x})$ to be a primitive polynomial and we consider $P(\mathsf{x})$ in the maximum case. In this case, N can be up to $2^{\delta n}-1$, the period of A. For larger integers, the sequence would be repeated. From now on we consider $N = 2^{\delta n}-1$

The next example helps us to understand this construction.

Example 4.4: Consider the primitive polynomial $p(\mathsf{x}) = 1+\mathsf{x}^3+\mathsf{x}^4$. We consider the polynomial matrix

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x} & 0 & 0 & 1 \\ 1 & \mathsf{x} & 0 & 0 \\ 0 & 1 & \mathsf{x}+1 & 0 \\ 0 & 0 & 1 & \mathsf{x} \end{bmatrix}$$

whose determinant is $p(\mathsf{x})$ and the corresponding matrix A is given by

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

In this example, $\delta = 1$ and $n = 4$. Then, the period of A is 15, therefore the matrix G in expression (4.7) has size 4×60, with $N = 2^4 - 1$, the maximum possible length. In this case, from a polynomial of degree 4 we get a sequence of length 60, four times longer than the sequence obtained from the same polynomial in the basic LFSR case. Therefore, we could encode a sequence four times longer. ■

A GLFSR has advantages over a basic LFSR. However, the sequence is still linear and, therefore, we can not use the sequence obtained from a generalized LFSR as a keystream. We have to use an approach as the one proposed in Figure 1.4, where several LFSRs are combined via a nonlinear Boolean function.

4.3.2 Correlation Problem

The correlation problem we present in this subsection was suggested by Professor Rosenthal [13].

Let $P(\mathsf{x})$ be a polynomial matrix described as in Subsection 4.3.1 and let the sequence $\boldsymbol{y} = [\ \boldsymbol{y}_0 \quad \boldsymbol{y}_1 \quad \cdots \quad \boldsymbol{y}_{N-1}\ ]$ be computed as

$$\left[\ \boldsymbol{y}_0 \quad \boldsymbol{y}_1 \quad \cdots \quad \boldsymbol{y}_{N-1}\ \right] = \boldsymbol{x}_0 G$$

where

$$G = \left[\ C \quad AC \quad \cdots \quad A^{N-1}C\ \right]$$

with A, C computed as in expression (4.8) and $\boldsymbol{x}_0$ the initial state.

Assume one receives the noisy sequence $\boldsymbol{z} = [\ \boldsymbol{z}_0 \quad \boldsymbol{z}_1 \quad \ldots \quad \boldsymbol{z}_{N-1}\ ]$, instead of the sequence $\boldsymbol{y}$. The problem now is to recover the initial state $\boldsymbol{x}_0$ (the key).

Let $\epsilon > 0.5$ be the correlation between $\boldsymbol{y}$ and $\boldsymbol{z}$. If the parity-check matrix of the code generated by G is sparse we can obtain the initial state just considering the problem as an LDPC decoding problem [37, 38].

In other cases, the problem is different. One can try brute force

$$\min_{x_0 \in \mathbb{F}_2^{\delta n}} \omega(\boldsymbol{x}_0 G - \boldsymbol{z})$$

where $\omega(\boldsymbol{x})$ is the Hamming weight of $\boldsymbol{x}$. Since the rows of G are a basis of the space of the possible sequences it is not necessary to compute the product $\boldsymbol{x}_0 G$ for all $\boldsymbol{x}_0 \in \mathbb{F}_2^{\delta n}$. The problem can be reduced to

$$\min_{\boldsymbol{x}_0 \in \mathbb{F}_2^{\delta n}} \omega\left(\sum_{i \in \ \mathrm{supp}(\boldsymbol{x}_0)} G_i - \boldsymbol{z}\right) \tag{4.9}$$

where $\mathrm{supp}(\boldsymbol{x})$ is the set of components of $\boldsymbol{x}$ which are not zero and G_i is the ith row of G.

If the sequence is highly noisy the situation is different. It is possible to find an initial state $\boldsymbol{x}_k$ which minimizes expression (4.9) and is not the desired state.

If we knew, for example, that the correlation $\epsilon < 0.5$ is small, then we can add 1 to all the bits of the sequence and the resultant sequence

$$\hat{\boldsymbol{z}} = \boldsymbol{z} + \mathbf{1}$$

is supposed to be correlated with $\boldsymbol{y}$ with correlation $1 - \epsilon > 0.5$. Then we can try to solve (4.9) with the new sequence $\hat{\boldsymbol{z}}$.

This problem is still an open problem. We might have reduced the complexity of the attack, but it keeps being a brute force attack.

4.3.3 Relation with Codes

As we know, we can consider G as the generator matrix of a code. In the basic LFSR case, we have a polynomial $p(\mathsf{x})$ of degree l and we consider that the matrix G in expression (4.7), has the maximum possible size $l \times (2^l - 1)$. The set of columns of G is equal to the set $\mathbb{F}_2^l \setminus \{\mathbf{0}\}$, so G is the generator matrix of a simplex code of order l.

Remember that a **simplex code** $S(t)$, or maximum length code, is the dual code of a Hamming code. The parameters of $S(t)$ are $[2^t - 1, t, 2^{t-1}]$. Moreover, the codewords have weight 2^{t-1}.

If we consider a primitive polynomial $p(\mathsf{x})$ of degree t we can obtain a polynomial matrix $P(\mathsf{x}) = \sum_{i=0}^{\delta-1} P_i \mathsf{x}^i + I\mathsf{x}^\delta \in \mathrm{Mat}_{n\times n}(\mathbb{F}_2[\mathsf{x}])$, with $\delta n = t$, such that $\det P(\mathsf{x}) = p(\mathsf{x})$. Given A and C constructed as in expression (4.8), we can construct G, see expression (4.7), of size $t \times n(2^t - 1)$. In this case, the columns of G are n copies of the nonzero elements of $\mathbb{F}_2^t$. Consequently, we can consider the matrix G as the generator matrix of a generalized Simplex code $S(t, n)$. The parameters are $[n \cdot (2^t - 1), t, n \cdot 2^{t-1}]$ and the codewords have weight $n \cdot (2^{t-1})$.

In order to understand this relationship, we provide the next example.

Example 4.5: Consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x}^3 + \mathsf{x}^4$, which is the determinant of the matrix

$$P(\mathsf{x}) = \begin{bmatrix} \mathsf{x}^2 + \mathsf{x} & 1 \\ 1 & \mathsf{x}^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \mathsf{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathsf{x}^2.$$

We can compute A and C as in expression (4.8),

$$A = \left[\begin{array}{cc:cc} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right] \quad \text{and} \quad C = \left[\begin{array}{c} 1 \quad 0 \\ 0 \quad 1 \\ \hdashline 0 \quad 0 \\ 0 \quad 0 \end{array}\right].$$

Since $\det P(\mathsf{x}) = 1 + \mathsf{x}^3 + \mathsf{x}^4$ is primitive of degree $t = \delta n = 4$ (with $\delta = 2$ and $n = 2$) the period of A is 15. Now, we can construct G as in expression (4.7):

$$G = \begin{bmatrix} C & AC & A^2C & \cdots & A^{13}C & A^{14}C \end{bmatrix} =$$

$$\left[\begin{array}{cccc:cccccccccccccccccccccccccc} 1&0&0&0&0&1&0&0&1&0&1&0&1&1&1&1&0&1&1&1&0&0&1&1&1&0&0&1&0&1\\ 0&1&0&0&1&0&1&0&1&1&1&1&0&1&1&1&0&0&1&1&1&0&0&1&0&1&1&0&0&0\\ 0&0&1&0&1&0&1&1&1&1&0&1&1&1&0&0&1&1&1&0&0&1&0&1&1&0&0&0&0&1\\ 0&0&0&1&0&0&1&0&1&0&1&1&1&1&0&1&1&1&0&0&1&1&1&0&0&1&0&1&1&0 \end{array}\right].$$

The set of columns of G is the set of element in $\mathbb{F}_2^4\backslash\{\mathbf{0}\}$, but each one appears twice. The matrix G is the generator matrix of a generalized Simplex code $S(4,2)$, with parameters $[30,4,16]$. ■

Consequenly, LFSRs and GLFSRs can be considered and studied as Simplex codes. However, this work is about codes over extension alphabets, so we study these codes as $\mathbb{F}_2$-linear codes.

We can consider the matrix G as the generator matrix of a cyclic $\mathbb{F}_2$-linear code over $\mathbb{F}_2^n$,

$$G = \left[\begin{array}{cccccc} C & AC & A^2C & \cdots & A^{N-1}C \end{array}\right]. \tag{4.10}$$

Assume now that $N = m\delta$ for some positive integer m. Then

$$\left[\begin{array}{cccccc} C & AC & A^2C & A^3C & \cdots & A^{\delta-1}C \end{array}\right] = I_{n\delta}$$

where $I_{n\delta}$ denotes the $\delta n \times \delta n$ identity matrix. Consequently

$$\begin{aligned}\left[\begin{array}{cccccc} A^\delta C & A^{\delta+1}C & A^{\delta+2}C & A^{\delta+3}C & \cdots & A^{2\delta-1}C \end{array}\right]\\ = A^\delta \left[\begin{array}{cccccc} C & AC & A^2C & A^3C & \cdots & A^{\delta-1}C \end{array}\right] = A^\delta\end{aligned}$$

and so on. So, we can write the generator matrix G of $\mathcal{C}_{\mathbb{F}_2^{\delta n}}$ as

$$G = \left[\begin{array}{c:ccccc} I_{n\delta} & A^\delta & A^{2\delta} & A^{3\delta} & \cdots & A^{(m-2)\delta} & A^{(m-1)\delta} \end{array}\right].$$

Since the generator matrix is in systematic form, the corresponding parity-check matrix is

$$H = \left[\begin{array}{c:c} \left(A^\delta\right)^T & \\ \left(A^{2\delta}\right)^T & \\ \left(A^{3\delta}\right)^T & \\ \vdots & I_{(m-1)n\delta} \\ \left(A^{(m-2)\delta}\right)^T & \\ \left(A^{(m-1)\delta}\right)^T & \end{array}\right].$$

This is the parity-check matrix of an $\mathbb{F}_2$-linear code $\mathcal{C}_{\mathbb{F}_2^{\delta n}}$ with length m, dimension 1 and minimum distance m. It is the same case we considered for basic LFSRs in Section 4.2, but instead of a code over $\mathbb{F}_2^l$, we have a code over $\mathbb{F}_2^{\delta n}$, where $n\delta$ is the maximum order for the polynomial matrix $P(\mathsf{x})$.

4.4 Constructing Primitive Polynomials

In Section 4.3.1 we use the polynomial matrix $P(\mathsf{x})$ in order to construct a generalized LFSR. In Theorem 4.3 we said that it is preferable to have a matrix whose determinant is a primitive polynomial. The question is: *given a primitive polynomial $p(\mathsf{x})$, how hard is to obtain a matrix $P(\mathsf{x})$, such that* $\det P(\mathsf{x}) = p(\mathsf{x})$*?*

Let $\mathbb{F}_q$ be the Galois field of q elements and consider the primitive polynomial

$$p(\mathsf{x}) = 1 + p_1\mathsf{x} + p_2\mathsf{x}^2 + \cdots + p_{l-1}\mathsf{x}^{l-1} + \mathsf{x}^l \in \mathbb{F}_q[\mathsf{x}].$$

We can use $p(\mathsf{x})$ in order to construct $\mathbb{F}_{q^l}$, the Galois field of q^l elements [31]. Thus,

$$\mathbb{F}_{q^l} \approx \mathbb{F}_q[\mathsf{x}]/\langle p(\mathsf{x})\rangle = \{q_{l-1}\mathsf{x}^{l-1} + q_{l-2}\mathsf{x}^{l-2} + \cdots + q_1\mathsf{x} + q_0 \mid q_i \in \mathbb{F}_q, i = 0, 1, \ldots, l-1\}$$

and addition and multiplication in $\mathbb{F}_{q^l}$ are the usual in $\mathbb{F}_q[\mathsf{x}]$, but reducing modulo $p(\mathsf{x})$.

Another way to see the elements in $\mathbb{F}_{q^l}$ is as vectors in $\mathbb{F}_q^l$, that is,

$$\mathbb{F}_{q^l} = \{[q_0, q_1, \ldots, q_{l-1}] \mid q_i \in \mathbb{F}_q\}$$

or as the q-ary representation of any integer in $\mathbb{F}_{q^l}$,

$$\mathbb{F}_{q^l} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots, \boldsymbol{q^l - 1}\}.$$

We can also see the elements in $\mathbb{F}_{q^l}$ as matrices. For this purpose, we consider the companion matrix P of the primitive polynomial $p(\mathsf{x})$,

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -p_{l-2} \\ 0 & 0 & \cdots & 1 & -p_{l-1} \end{bmatrix}.$$

Now, for a primitive element $\alpha \in \mathbb{F}_{q^l}$ we consider the isomorphism $\psi : \mathbb{F}_{q^l} \to \mathbb{F}_q[P]$, such that $\psi(\alpha) = P$, introduced in Theorem 3.3. Then we can write

$$\mathbb{F}_{q^l} = \{O, I, P, P^2, \ldots, P^{q^l-2}\}.$$

As we saw in Section 3.3, this isomorphism can be extended to a ring isomorphism $\Psi : \mathrm{Mat}_{t\times m}(\mathbb{F}_{q^l}) \to \mathrm{Mat}_{t\times m}(\mathbb{F}_q[P])$, such that $\Psi(M) = [\psi(m_{i,j})]$, for $M = [m_{i,j}]$. The following theorem shows that this ring ismorphism helps us to construct primitive polynomials with coefficients in $\mathbb{F}_q$ using primitive polynomials with coefficients in $\mathbb{F}_{q^l}$.

Theorem 4.4: *If Q is the companion matrix of a primitive polynomial $q(\mathsf{x}) \in \mathbb{F}_{q^l}[\mathsf{x}]$ of degree r, then the characteristical polynomial of the matrix $M = \Psi(Q)$, where Ψ is the ring isomorphism given in Theorem 3.3, is a primitive polynomial $m(\mathsf{x}) \in \mathbb{F}_q[\mathsf{x}]$ of degree rl. That is, $m(\mathsf{x}) = \det(\mathsf{x}I - M)$ is a primitive polynomial of degree rl with coefficients in $\mathbb{F}_q$.*

PROOF: Remember that the field isomorphism we constructed in Theorem 3.3 was given by $\psi : \mathbb{F}_{q^l} \to \mathbb{F}_q[P]$, such that, $\psi(\alpha) = P$, where $\alpha \in \mathbb{F}_{q^l}$ is a primitive element, and P is the companion matrix of a primitive polynomial $p(\mathsf{x})$ of degree l in $\mathbb{F}_q[\mathsf{x}]$.

From this field isomorphism, we can construct the ring isomorphism, given also in Theorem 3.3, $\Psi : \mathrm{Mat}_{t\times m}(\mathbb{F}_{q^l}) \to \mathrm{Mat}_{t\times m}(\mathbb{F}_q[P])$, such that, $\Psi(M) = [\psi(M_{ij})]$.

Consider the primitive polynomial $q(\mathsf{x}) = q_0 + q_1\mathsf{x} + \cdots + q_{r-1}\mathsf{x}^{r-1} + \mathsf{x}^r \in \mathbb{F}_{q^l}[\mathsf{x}]$, whose companion matrix is

$$Q = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -q_0 \\ 1 & 0 & 0 & \cdots & 0 & -q_1 \\ 0 & 1 & 0 & \cdots & 0 & -q_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -q_{r-2} \\ 0 & 0 & 0 & \cdots & 1 & -q_{r-1} \end{bmatrix}.$$

We have to prove that the characteristic polynomial of the matrix M given by

$$M = \Psi(Q) = \begin{bmatrix} O & O & O & \cdots & O & \psi(-q_0) \\ I_l & O & O & \cdots & O & \psi(-q_1) \\ O & I_l & O & \cdots & O & \psi(-q_2) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ O & O & O & \cdots & O & \psi(-q_{r-2}) \\ O & O & O & \cdots & I_l & \psi(-q_{r-1}) \end{bmatrix}$$

is a primitive polynomial in $\mathbb{F}_q[\mathsf{x}]$ with degree rl.

The matrix M has size $rl \times rl$, so the characteristic polynomial $m(\mathsf{x}) = \det(\mathsf{x}I_{rl} - \Psi(Q))$ has degree rl.

The polynomial $m(\mathsf{x})$ is a primitive polynomial if and only if all the powers $\Psi(Q)^i$, for $i = 1, 2, \ldots, q^{rl} - 1$ are pairwise different.

First of all, $\Psi(Q)^i \in \mathrm{Mat}_{rl\times rl}(\mathbb{F}_q)$, for $i = 1, 2, \ldots, q^{rl} - 1$. Suppose $\Psi(Q)^i = \Psi(Q)^j$, for $i, j \in \{1, 2, \ldots, q^{rl} - 1\}$. Since Ψ is an isomorphism, we have

$$\Psi(Q^i) = \Psi(Q^j)$$

and since it is bijective

$$Q^i = Q^j \quad \text{in} \quad \mathrm{Mat}_{r\times r}(\mathbb{F}_{q^l}).$$

The matrix Q is the companion matrix of a primitive polynomial in $\mathbb{F}_{q^l}[\mathsf{x}]$, then $\mathbb{F}_{q^{lr}} \approx \mathbb{F}_{q^l}[Q]$, therefore

$$i - j \equiv 0 \pmod{(q^l)^r - 1},$$

that is,

$$i = j + t(q^{rl} - 1).$$

Since $i, j \in \{1, 2, \ldots, q^{rl} - 1\}$, necessarily $t = 0$. This means that $i = j$ and the matrices $\Psi(Q)^i$, for $i = 1, 2, \ldots, q^{rl} - 1$, are pairwise different. □

If we need a polynomial matrix whose determinant is $m(\mathsf{x})$ we can choose either $\mathsf{x}I - M$ (of size $rl \times rl$) or $\sum_{i=0}^{r-1} \psi(-q_i)\mathsf{x}^i + I_l\mathsf{x}^r$ (of size $l \times l$).

The next example illustrates this construction.

Example 4.6: Consider the primitive polynomial $p(\mathsf{x}) = 1 + \mathsf{x} + \mathsf{x}^2 \in \mathbb{F}_2[\mathsf{x}]$. We can use this polynomial to construct the Galois field $\mathbb{F}_{2^2}$ of 4 elements. Thus,

$$\mathbb{F}_{2^2} \approx \mathbb{F}_2[\mathsf{x}]/\langle p(\mathsf{x})\rangle = \{a_1\mathsf{x} + a_0 \mid a_0, a_1 \in \mathbb{F}_2\}$$

and the addition and multiplication in $\mathbb{F}_{2^2}$ are the usual in $\mathbb{F}_2[\mathsf{x}]$ but modulo $p(\mathsf{x})$.

Another way to see the elements in $\mathbb{F}_{2^2}$ is as vectors in $\mathbb{F}_2^2$, that is

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

or,

$$\mathbb{F}_{2^2} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$$

with the operations given by the tables:

$+$	**0**	**1**	**2**	**3**
0	**0**	**1**	**2**	**3**
1	**1**	**0**	**3**	**2**
2	**2**	**3**	**0**	**1**
3	**3**	**2**	**1**	**0**

$\cdot$	**0**	**1**	**2**	**3**
0	**0**	**0**	**0**	**0**
1	**0**	**1**	**2**	**3**
2	**0**	**2**	**3**	**1**
3	**0**	**3**	**1**	**2**

Another way to see the elements in $\mathbb{F}_{2^2}$ is as matrices. For this purpose, we consider the companion matrix of $p(\mathsf{x})$, that is,

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

and, since $p(\mathsf{x})$ is a primitive polynomial, we know that

$$\mathbb{F}_{2^2} \approx \mathbb{F}_2[P] = \{O, I, P, P + I\} = \{O, I, P, P^2\}.$$

Besides, given $\alpha = \mathbf{3} \in \mathbb{F}_{2^2}$ a primitive element according to Theorem 3.3 we know that the map $\psi : \mathbb{F}_{2^2} \longrightarrow \mathbb{F}_2[P]$ such that $\psi(\mathbf{3}) = P$ is a field isomorphism. This isomorphism can be extended to a ring isomorphism $\Psi : \mathcal{M}_{t\times m}(\mathbb{F}_{2^2}) \to \mathcal{M}_{t\times m}(\mathbb{F}_2[P])$, such that $\Psi(M) = [\psi(M_{ij})]$, for $M \in \mathrm{Mat}_{t\times m}(\mathbb{F}_{2^2})$.

Consider the primitive polynomial $q(\mathsf{x}) = \mathbf{3} + \mathbf{2}\mathsf{x} + \mathsf{x}^2 + \mathsf{x}^3 \in \mathbb{F}_{2^2}[\mathsf{x}]$. The corresponding companion matrix is

$$Q = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{3} \\ \mathbf{1} & \mathbf{0} & \mathbf{2} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}.$$

Then, we can apply the ring isomorphism to the previous matrix

$$M = \Psi(Q) = \begin{bmatrix} O & O & P \\ I & O & P^2 \\ O & I & I \end{bmatrix} = \left[\begin{array}{cc:cc:cc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ \hdashline 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right].$$

The characteristic polynomial of this matrix

$$m(\mathsf{x}) = \det\left(\mathsf{x}I - \Psi(Q)\right) = 1 + \mathsf{x} + \mathsf{x}^6$$

is a primitive polynomial with coefficients in $\mathbb{F}_2$ and degree 6.

Summarizing, with a polynomial of degree 3 over $\mathbb{F}_{2^2}[\mathsf{x}]$ and a polynomial of degree 2 over $\mathbb{F}_2[\mathsf{x}]$ we found a polynomial $m(\mathsf{x})$ of degree 6 over $\mathbb{F}_2[\mathsf{x}]$. Now, if we had a polynomial of degree t over $\mathbb{F}_{2^6}$ and $m(\mathsf{x})$, we could find a polynomial of degree $6t$ over $\mathbb{F}_2[\mathsf{x}]$.

If we need a polynomial matrix whose determinant is the primitive polynomial $m(x)$, we can consider the matrix $P_1(\mathsf{x}) = P + P^2\mathsf{x} + I\mathsf{x}^2 + I\mathsf{x}^3$ (using the matrices in the last block column of M), or the matrix $P_2(\mathsf{x}) = \mathsf{x}I - M$. The orders of $P_1(\mathsf{x})$ and $P_2(\mathsf{x})$ are 2 and 6, respectively. If we want our sequence to be as long as possible, we should choose $P_2(\mathsf{x})$. ■

Appendix A

Conclusions

In Chapter 2 we propose the construction of $\mathbb{F}_2$-linear codes over $\mathbb{F}_2^b$ which are cyclic with low density. Actually, we construct the index array representing the parity-check matrix of the code. This array stores the position of each and every 1 in that matrix. We consider a partition of $\mathbb{F}_{p-1}$, with p a prime odd number. We apply the Zech logarithm to each element of these sets, and then we obtain the 0th colum of our index array. As we said before, these codes are special, they are cyclic and have low density. However, they are not always MDS. When the redundancy is $r = 2$, they seem to be MDS, but they are not MDS in other cases. This fact is based on the properties of the Zech logarithm, but this logarithm is a discrete logarithm, so it is not easy to find an easy proof. Besides, we propose a decoding algorithm based on the syndrome. It is computed using the index array, without even computing the parity-check matrix. This algorithm could be extended for another kind of codes, if we had the index array.

In Chapter 3 we propose another construction of $\mathbb{F}_q$-linear codes which are MDS over $\mathbb{F}_q^b$. If the parity-check matrix H of a code over $\mathbb{F}_q$ (respectively, generator matrix) is in systematic form $H = [\ I \quad A\ ]$ and the matrix A is superregular, then the code is MDS. We can find an extension of this result in [7]: if the parity-check matrix of an $\mathbb{F}_q$-linear code over $\mathbb{F}_q^b$ is in systematic form, and the matrix A is a b-block superregular matrix, then the code is MDS over $\mathbb{F}_q^b$. Then, our aim is to construct superregular b-block matrices over $\mathbb{F}_q$.

Given the companion matrix C of a primitive polynomial of degree b with coefficients in $\mathbb{F}_q$, we consider an isomorphism ψ between $\mathbb{F}_q^b$ and $\mathbb{F}_q[C]$, given by $\psi(\alpha) = C$, where $\alpha \in \mathbb{F}_q^b$ is a primitive element. If we have a superregular matrix of size $t \times m$, then we can construct a b-block superregular matrix of size $bt \times bm$ over $\mathbb{F}_q$ using the previous isomorphism.

We also propose a decoding algorithm based on the idea given in [3]. We consider each symbol in a codeword as polynomials of degree less or equal to b, and we consider the

preimage of H with the previous isomorphism instead of using the parity-check matrix H. Then we compute the syndrome and depending on the nonzero entries, we can detect and correct or not the errors.

In Chapter 4 we study LFSRs as codes. An LFSR can be expressed as an autonomous system. Furthermore, the observability matrix of the system can be seen as the generator matrix of an $\mathbb{F}_2$-linear code. Then, we propose a decoding algorithm similar to the one proposed in Chapter 3.

We also present the idea of generalized LFSRs. Instead of using a polynomial to construct the sequences, we use a polynomial matrix. In this case, we can also see this generalized LFSR as a code. These codes can be seen as $\mathbb{F}_2$-linear codes as we did with the basic LFSRs or they can be considered as Simplex codes.

Finally we introduce a method to construct primitive polynomials from other primitive polynomials of smaller degree and we indicate how to obtain a polynomial matrix given its determinat.

As future work, we want to keep working on the construction proposed in Chapter 2 in order to prove why the codes are MDS in some cases and not in others. Besides, we would like to extend the construction and the decoding algorithm for codes over $\mathbb{F}_q$, with $q > 2$. We think we could use the cyclicity and the low density property ir order to improve our algorithm or find a better decoding algorithm.

Professor Greferath suggested us to extend the construction in Chapter 3 for rings, specially for $\mathbb{Z}_4$. We have not obtained satisfactory results at the moment, but we would like to keep working on this idea. We also have to improve the algorithm, since the complexity grows exponentially when the length of the code grows.

Finally, GLFSRs can be used to construct keystream generators obtaining longer sequences than in the basic LFSR case. We are working on the construction of a keystream generator using GLFSRs and using different Boolean functions with 'good' cryptographic properties. We are testing for different functions if the sequences obtained are good for cryptographic use.

Bibliography

[1] M. Blaum, J. Brady, J. Bruck and J. Menon. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, **42(2)**: 192–202 (1995). xxi, 13, 38

[2] M. Blaum and J. Bruck. MDS array codes for correcting a single criss-cross error. *IEEE Transactions on Information Theory*, **46(3)**: 1068–1077 (2000). xxi

[3] M. Blaum, J. Bruck and A. Vardy. MDS array codes with independent parity symbols. *IEEE Transactions on Information Theory*, **42(2)**: 529–542 (1996). xxi, xxviii, 13, 58, 95

[4] M. Blaum, J. L. Fan and L. Xu. Soft decoding of several classes of array codes. Extended version of [5]. xxi

[5] M. Blaum, J. L. Fan and L. Xu. Soft decoding of several classes of array codes. In *Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT 2002)*, page 368. IEEE, Lausanne, Switzerland, July 2002. 13, 50, 97

[6] M. Blaum, P. G. Farrell and H. C. A. van Tilborg. Array codes. In V. S. Pless and W. C. Huffman (editors), *Handbook of Coding Theory*, pages 1855–1909. Elsevier, North-Holland, 1998. xxi, 13

[7] M. Blaum and R. M. Roth. On lowest density MDS codes. *IEEE Transactions on Information Theory*, **45(1)**: 46–59 (1999). xxi, xxviii, 2, 10, 11, 13, 15, 95

[8] J. A. Buhmann. *Introduction to Cryptography*. Springer, New York, NY, 2002. xxii

[9] P. Camion, M. J. Mihaljević and H. Imai. Two alerts for design of certain stream ciphers: Trapped LFSR and weak resilient function over $gf(q)$. In K. Nyberg and H. Heys (editors), *Selected Areas in Cryptography – SAC 2003*, volume 2595 of *Lecture Notes in Computer Science*, pages 196–213. Springer-Verlag, Berlin, 2003. xxii

[10] S. D. Cardell, J.-J. Climent and V. Requena. A construction of MDS array codes based on companion matrices. In J. Borges and M. Villanueva (editors), *Proceedings of the 3rd International Castle Meeting on Coding Theory and Applications*, pages 87–91. Barcelona, 2011. 47

[11] S. D. Cardell, J.-J. Climent and V. Requena. MDS array codes based on superregular matrices. In J. Vigo Aguiar (editor), *Proceedings of the 11th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2011)*, pages 290–295. 2011. 47

[12] S. D. Cardell, J.-J. Climent and R. Verónica. On the construction and decoding of $\mathbb{F}_q$-linear MDS codes. *Submitted*, (2012). 47

[13] S. D. Cardell, G. Maze, J. Rosenthal and U. Wagner. Correlations in stream ciphers: A systems theory point of view. In A. Edelmayer (editor), *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS2010)*, pages 419–423. 2010. 85

[14] V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In D. W. Davies (editor), *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 68–70. Springer-Verlag, Berlin, 1991. 23

[15] J.-J. Climent, D. Napp, C. Perea and R. Pinto. A construction of MDS 2D convolutional codes of rate $1/n$ based on superregular matrices. *Linear Algebra and its Applications*, **437**: 766–780 (2012). 47

[16] T. W. Cusick and P. Stănică. *Cryptographic Boolean Functions and Applications*. Academic Press, San Diego, CA, 2009. 19

[17] J. F. Dillon. Multiplicative difference sets via additive characters. *Designs, Codes and Cryptography*, **17**: 225–235 (1999). xxii

[18] P. Ekdhal. *On LFSR Based Stream Ciphers. Analysis and Design*. PhD Thesis, Department of Information Technology, Lund University, Lund, Sweden, October 2003. xxiii, 17, 23, 73

[19] M. Elia. On tridiagonal binary matrices and LFSRs. A survey. *Contemporary Engineering Sciences*, **3(4)**: 167–182 (2010). xxiii, 17, 73

[20] M. Fiedler. A note on companion matrices. *Linear Algebra and its Applications*, **372**: 325–331 (2003). 82

[21] R. G. Gallager. *Low-Density Parity-Check Codes*. Number 21 in Research Monograph Series. MIT Press, Cambridge, MA, 1963. xx, 9, 23

[22] D. B. Güçlü. *Cryptographic Properties of Some Highly Nonlinear Balanced Boolean Functions*. Master Thesis, Department of Cryptography, The Middle East Technical University, January 2006. xxiv

[23] K. Huber. Some comments on zech's logarithms. *IEEE Transactions on Information Theory*, **36(4)**: 946–950 (1990). 27, 31

[24] W. C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, New York, NY, 2003. 1, 2, 5, 6, 8

[25] R. Hutchinson, R. Smarandache and J. Trumpf. On superregular matrices and MDP convolutional codes. *Linear Algebra and its Applications*, **428**: 2585–2596 (2008). 47, 52, 53

[26] R. Johannesson and K. S. Zigangirov. *Fundamentals of Convolutional Coding*. IEEE Press, New York, NY, 1999. xx

[27] T. Johansson and F. Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In J. Stern (editor), *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362. Springer-Verlag, Berlin, 1999. 23

[28] S. J. Johnson. *Iterative Error Correction*. Cambridge University Press, Cambridge, UK, 2010. xx

[29] G. Kéri. Types of superregular matrices and the number of n-arcs and complete n-arcs in $PG(r, q)$. *Journal of Combinatorial Designs*, **14(5)**: 363–390 (2006). xxvii, 47, 54

[30] J. Lacan and J. Fimes. A construction of matrices with no singular square submatrices. In G. L. Mullen, A. Poli and H. Stichtenoth (editors), *Finite Fields and Applications*, volume 2948 of *Lecture Notes in Computer Science*, pages 145–147. Springer-Verlag, Berlin, 2003. xxvii, 56

[31] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, New York, NY, 1994. xxiii, xxviii, 19, 48, 89

[32] S. Lin and D. J. Costello, Jr. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983. xix

[33] E. Louidor and R. M. Roth. Lowest density MDS codes over extension alphabets. *IEEE Transactions on Information Theory*, **52(7)**: 46–59 (2006). xxi, 2, 10, 13

[34] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 6th edition, 1988. xix, xx, xxvii, 2, 5, 6, 8, 15

[35] J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, **15(1)**: 122–127 (1969). xxiv, 19, 73

[36] R. J. McEliece. The algebraic theory of convolutional codes. In V. S. Pless and W. C. Huffman (editors), *Handbook of Coding Theory*, pages 1065–1138. Elsevier, North-Holland, 1998. xix

[37] W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In C. G. Günter (editor), *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer-Verlag, Berlin, 1988. 22, 23, 86

[38] W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, **1(3)**: 159–176 (1989). 22, 23, 86

[39] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1996. xxii, xxiii, 2, 3

[40] H. Molland, J. E. Mathiassen and T. Helleseth. Improved fast correlation attack using low rate codes. In K. G. Paterson (editor), *Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 67–81. Springer-Verlag, New York, NY, 2003. 23

[41] S. Myung, K. Yang and J. Kim. Quasi-cyclic LDPC codes for fast encoding. *IEEE Transactions on Information Theory*, **51(8)**: 2894–2901 (2005). 9

[42] D. Olejár and M. Stanek. On cryptographic properties of random Boolean functions. *Journal of Universal Computer Science*, **4(8)**: 705–717 (1998). xxiv

[43] T. Richardon and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2007. xx

[44] R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, **21(2)**: 120–126 (1978). xxii

[45] S. Roman. *Coding and Information Theory*. Springer, New York, NY, 1992. xix, 6, 7

[46] J. Rosenthal. Connections between linear systems and convolutional codes. In B. Marcus and J. Rosenthal (editors), *Codes, Systems and Graphical Models*, volume 123 of *The IMA Volumes in Mathematics and its Applications*, pages 39–66. Springer-Verlag, New York, 2001. xix

[47] R. M. Roth and A. Lempel. On MDS codes via Cauchy matrices. *IEEE Transactions on Information Theory*, **35(6)**: 1314–1319 (1989). xxvii, 47, 55

[48] R. M. Roth and G. Seroussi. On generator matrices of MDS codes. *IEEE Transactions on Information Theory*, **31(6)**: 826–830 (1985). 47, 54, 55

[49] W. E. Ryan. An introduction to LDPC codes. In B. Vasic and E. M. Kurtas (editors), *CRC Handbook for Coding and Signal Processing for Recoding Systems*. CRC Press, 2004. 23

[50] P. Sarkar and S. Maitra. Construction of nonlinear Boolean functions with important cryptographic properties. In B. Preneel (editor), *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 485–506. Springer-Verlag, Berlin, 2000. xxiv

[51] B. Schneier. *Applied Cryptography*. John Wiley & Sons, New York, NY, second edition, 1996. xxiii, 2, 3

[52] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, **27**: 379–423 (1948). 1

[53] A. Shokrollahi. LDPC codes: An introduction. In C. Coding and Combinatorics (editors), *Computer Science – Theory and Applications*, volume 23 of *Progress in Computer Science and Applied Logic*, pages 85–110. Birkhauser, Basel, 2004. 9

[54] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, **30(5)**: 776–780 (1984). 21

[55] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, **34(1)**: 81–85 (1985). 21

[56] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja and D. J. Costello, Jr. LDPC block and convolutional codes based on circulant matrices. *IEEE Transactions on Information Theory*, **50(12)**: 2966–2984 (2004). 9

[57] V. Tomás, J. Rosenthal and R. Smarandache. Decoding of convolutional codes over the erasure channel. *IEEE Transactions on Information Theory*, **58(1)**: 90–108 (2012). 47, 52, 53

[58] U. Wagner. *Detection and Exploitation of Small Correlations in Stream Ciphers*. Diploma thesis, Institut für Mathematics, Universität Zürich, Zürich, Switzerland, 2008. xxiii, xxiv, 17, 22, 73

[59] J. Xu, L. Chen, I. Djurdjevic, S. Lin and K. Abdel-Ghaffar. Construction of regular and irregular LDPC codes: geometry decomposition and masking. *IEEE Transactions on Information Theory*, **53(1)**: 121–134 (2007). 9

[60] L. Xu, V. Bohossian, J. Bruck and D. G. Wagner. Low-density MDS codes and factors of complete graphs. *IEEE Transactions on Information Theory*, **45(6)**: 1817–1826 (1999). xxi, 13

[61] L. Xu and J. Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, **45(1)**: 272–276 (1999). xxi, 13

[62] K. Yang and T. Helleseth. On the minimum distance of array codes as LDPC codes. *IEEE Transactions on Information Theory*, **49(12)**: 3268–3271 (2003). 9