2.1 Finite Fields

In this section we present a basic introduction to the theory of finite fields. For a more formal and elaborate introduction, we refer to the work of Lidl and Niederreiter [58].

2.1.1 Groups, Rings, and Fields

We start with the formal definition of a group.

Definition 2.1.1. *An* Abelian group $< G, + >$ *consists of a set* G *and an operation defined on its elements, here denoted by '+':*

$$+ : G \times G \to G : (a, b) \mapsto a + b. \tag{2.1}$$

In order to qualify as an Abelian group, *the operation has to fulfill the following conditions:*

$$\begin{aligned}
\textit{closed:} &\ \forall\ a, b \in G : a + b \in G && (2.2)\\
\textit{associative:} &\ \forall\ a, b, c \in G : (a + b) + c = a + (b + c) && (2.3)\\
\textit{commutative:} &\ \forall\ a, b \in G : a + b = b + a && (2.4)\\
\textit{neutral element:} &\ \exists\, \mathbf{0} \in G, \forall\ a \in G : a + \mathbf{0} = a && (2.5)\\
\textit{inverse elements:} &\ \forall\ a \in G, \exists\ b \in G : a + b = \mathbf{0} && (2.6)
\end{aligned}$$

Example 2.1.1. The best-known example of an Abelian group is $< \mathbb{Z}, + >$: the set of integers, with the operation 'addition'. The structure $< \mathbb{Z}_n, + >$ is a second example. The set contains the integer numbers 0 to $n - 1$ and the operation is addition modulo n.

Since the addition of integers is the best known example of a group, usually the symbol '+' is used to denote an arbitrary group operation. In this book, both an arbitrary group operation and integer addition will be denoted by the symbol '+'. For some special types of groups, we will denote the addition operation by the symbol '$\oplus$' (see Sect. 2.1.3).

Both rings and fields are formally defined as structures that consist of a set of elements with *two* operations defined on these elements.

Definition 2.1.2. *A* ring $< R, +, \cdot >$ *consists of a set* R *with two operations defined on its elements, here denoted by '+' and '·'. In order to qualify as a ring, the operations have to fulfill the following conditions:*

1. *The structure* $< R, + >$ *is an Abelian group.*
2. *The operation '·' is closed, and associative over* R. *There is a neutral element for '·' in* R.

3. *The two operations '+' and '·' are related by the law of distributivity:*

$$\forall\ a, b, c \in R:\ (a + b) \cdot c = (a \cdot c) + (b \cdot c). \tag{2.7}$$

The neutral element for '·' is usually denoted by **1**. A ring $< R, +, \cdot >$ is called a *commutative ring* if the operation '·' is commutative.

Example 2.1.2. The best-known example of a ring is $< \mathbb{Z}, +, \cdot >$: the set of integers, with the operations 'addition' and 'multiplication'. This ring is a commutative ring. The set of matrices with n rows and n columns, with the operations 'matrix addition' and 'matrix multiplication' is a ring, but not a commutative ring (if $n > 1$).

Definition 2.1.3. *A structure* $< F, +, \cdot >$ *is a* field *if the following two conditions are satisfied:*

1. $< F, +, \cdot >$ *is a commutative ring.*
2. *For all elements of* F*, there is an inverse element in* F *with respect to the operation '·', except for the element* **0***, the neutral element of* $< F, + >$.

A structure $< F, +, \cdot >$ is a field iff both $< F, + >$ and $< F \backslash \{\mathbf{0}\}, \cdot >$ are Abelian groups and the law of distributivity applies. The neutral element of $< F \backslash \{\mathbf{0}\}, \cdot >$ is called the *unit element* of the field.

Example 2.1.3. The best-known example of a field is the set of real numbers, with the operations 'addition' and 'multiplication.' Other examples are the set of complex numbers and the set of rational numbers, with the same operations. Note that for these examples the number of elements is infinite.

2.1.2 Vector Spaces

Let $< F, +, \cdot >$ be a field, with unit element **1**, and let $< V, + >$ be an Abelian group. Let '$\odot$' be an operation on elements of F and V:

$$\odot:\ F \times V \to V. \tag{2.8}$$

Definition 2.1.4. *The structure* $< F, V, +, +, \cdot, \odot >$ *is a* vector space *over* F *if the following conditions are satisfied:*

1. *distributivity:*

$$\forall\ a \in F, \forall\ \mathbf{v}, \mathbf{w} \in V:\ a \odot (\mathbf{v} + \mathbf{w}) = (a \odot \mathbf{v}) + (a \odot \mathbf{w}) \tag{2.9}$$

$$\forall\ a, b \in F, \forall\ \mathbf{v} \in V:\ (a + b) \odot \mathbf{v} = (a \odot \mathbf{v}) + (a \odot \mathbf{v}) \tag{2.10}$$

2. *associativity:*

$$\forall\ a, b \in F, \forall\ \mathbf{v} \in V: (a \cdot b) \odot \mathbf{v} = a \odot (b \odot \mathbf{v}) \tag{2.11}$$

3. neutral element:

$$\forall\ \mathbf{v} \in V : \mathbf{1} \odot \mathbf{v} = \mathbf{v}. \tag{2.12}$$

The elements of V are called *vectors*, and the elements of F are the *scalars*. The operation '$+$' is called the *vector addition*, and '$\odot$' is the *scalar multiplication*.

Example 2.1.4. For any field F, the set of n-tuples $(a_0, a_1, \ldots, a_{n-1})$ forms a vector space, where '$+$' and '$\odot$' are defined in terms of the field operations:

$$(a_1, \ldots, a_n) + (b_1, \ldots, b_n) = (a_1 + b_1, \ldots, a_n + b_n) \tag{2.13}$$

$$a \odot (b_1, \ldots, b_n) = (a \cdot b_1, \ldots, a \cdot b_n). \tag{2.14}$$

A vector $\mathbf{v}$ is a *linear combination* of the vectors $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(s)}$ if there exist scalars $a^{(i)}$ such that:

$$\mathbf{v} = a^{(1)} \odot \mathbf{w}^{(1)} + a^{(2)} \odot \mathbf{w}^{(2)} + \cdots + a^{(s)} \odot \mathbf{w}^{(s)}. \tag{2.15}$$

In a vector space we can always find a set of vectors such that all elements of the vector space can be written in exactly one way as a linear combination of the vectors of the set. Such a set is called a *basis* of the vector space. We will consider only vector spaces where the bases have a finite number of elements. We denote a basis by

$$\mathsf{e} = \left[\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \ldots \mathbf{e}^{(n)}\right]^{\mathrm{T}}. \tag{2.16}$$

In this expression the T superscript denotes taking te transpose of the column vector e. The scalars used in this linear combination are called the *coordinates* of $\mathbf{x}$ with respect to the basis e:

$$\mathrm{co}(\mathbf{x}) = \boldsymbol{x} = (c_1, c_2, \ldots, c_n) \Leftrightarrow \mathbf{x} = \textstyle\sum_{i=1}^{n} c_i \odot \mathbf{e}^{(i)}. \tag{2.17}$$

In order to simplify the notation, from now on we will denote vector addition by the same symbol as the field addition ('$+$'), and the scalar multiplication by the same symbol as the field multiplication ('$\cdot$'). It should always be clear from the context what operation the symbols are referring to.

A function f is called *a linear function* of a vector space V over a field F, if it has the following properties:

$$\forall\ \mathbf{x}, \mathbf{y} \in V :\ f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y}) \tag{2.18}$$

$$\forall\ a \in F, \forall\ \mathbf{x} \in V :\ f(a\mathbf{x}) = af(\mathbf{x}). \tag{2.19}$$

The linear functions of a vector space $\mathrm{GF}(p)^n$ iff there exists a matrix M such that

$$\mathrm{co}(f(\mathbf{x})) = \mathsf{M} \cdot \boldsymbol{x}, \forall\ \mathbf{x} \in \mathrm{GF}(p)^n. \tag{2.20}$$

2.1.3 Fields with a Finite Number of Elements

A *finite field* is a field with a finite number of elements. The number of elements in the set is called the *order* of the field. A field with order m exists iff m is a *prime power*, i.e. $m = p^n$ for some integer n and with p a prime integer. p is called the *characteristic* of the finite field.

All finite fields used in the description of Rijndael have a characteristic of 2. By the symbol '$\oplus$', we will always denote the addition operation in a field with a characteristic of 2.

Fields of the same order are *isomorphic*: they display exactly the same algebraic structure differing only in the *representation* of the elements. In other words, for each prime power there is exactly one finite field, denoted by $\mathrm{GF}(p^n)$. From now on, we will only consider fields with a finite number of elements.

Perhaps the most intuitive examples of finite fields are the fields of prime order p. The elements of a finite field $\mathrm{GF}(p)$ can be represented by the integers $0, 1, \ldots, p-1$. The two operations of the field are then 'integer addition modulo p' and 'integer multiplication modulo p'.

For finite fields with an order that is not prime, the operations *addition* and *multiplication* cannot be represented by addition and multiplication of integers modulo a number. Instead, slightly more complex representations must be introduced. Finite fields $\mathrm{GF}(p^n)$ with $n > 1$ can be represented in several ways. The representation of $\mathrm{GF}(p^n)$ by means of polynomials over $\mathrm{GF}(p)$ is quite popular and is the one we have adopted in Rijndael and its predecessors. In the next sections, we explain this representation.

2.1.4 Polynomials over a Field

A polynomial over a field F is an expression of the form

$$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_2x^2 + b_1x + b_0, \tag{2.21}$$

x being called the *indeterminate* of the polynomial, and the $b_i \in F$ the *coefficients*.

We will consider polynomials as abstract entities only, which are never evaluated. Because the sum is never evaluated, we always use the symbol '+' in polynomials, even if they are defined over a field with characteristic 2.

The *degree* of a polynomial equals ℓ if $b_j = 0, \forall j > \ell$, and ℓ is the smallest number with this property. The set of polynomials over a field F is denoted by $F[x]$. The set of polynomials over a field F, which have a degree below ℓ, is denoted by $F[x]|_\ell$.

In computer memory, the polynomials in $F[x]|_\ell$ with F a finite field can be stored efficiently by storing the ℓ coefficients as a string.

Example 2.1.5. Let the field F be GF(2), and let $\ell = 8$. The polynomials can conveniently be stored as 8-bit values, or bytes:

$$b(x) \mapsto b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0. \tag{2.22}$$

Strings of bits are often abbreviated using the hexadecimal notation.

Example 2.1.6. The polynomial in $\mathrm{GF}(2)|_8$

$$x^6 + x^4 + x^2 + x + 1$$

corresponds to the bit string 01010111, or `57` in hexadecimal notation.

2.1.5 Operations on Polynomials

We define the following operations on polynomials.

Addition. Summing of polynomials consists of summing the coefficients with equal powers of x, where the summing of the coefficients occurs in the underlying field F:

$$c(x) = a(x) + b(x) \Leftrightarrow c_i = a_i + b_i, \quad 0 \leq i < n. \tag{2.23}$$

The neutral element for the addition $\mathbf{0}$ is the polynomial with all coefficients equal to 0. The inverse element of a polynomial can be found by replacing each coefficient by its inverse element in F. The degree of $c(x)$ is at most the maximum of the degrees of $a(x)$ and $b(x)$, hence the addition is closed. The structure $< F[x]|_\ell, + >$ is an Abelian group.

Example 2.1.7. Let F be the field GF(2). The sum of the polynomials denoted by `57` and `83` is the polynomial denoted by `D4`, since:

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1) &\oplus (x^7 + x + 1) \\ &= x^7 + x^6 + x^4 + x^2 + (1 \oplus 1)x + (1 \oplus 1) \\ &= x^7 + x^6 + x^4 + x^2.\end{aligned}$$

In binary notation we have: $01010111 \oplus 10000011 = 11010100$. Clearly, the addition can be implemented with the bitwise XOR instruction.

Multiplication. Multiplication of polynomials is associative (2.3), commutative (2.4) and distributive (2.7) with respect to addition of polynomials. There is a neutral element: the polynomial of degree 0 and with coefficient of x^0 equal to 1. In order to make the multiplication closed (2.2) over $F[x]|_\ell$, we select a polynomial $m(x)$ of degree ℓ, called the *reduction polynomial.*

The multiplication of two polynomials $a(x)$ and $b(x)$ is then defined as the algebraic product of the polynomials modulo the polynomial $m(x)$:

$$c(x) = a(x) \cdot b(x) \Leftrightarrow c(x) \equiv a(x) \times b(x) \pmod{m(x)}. \tag{2.24}$$

Hence, the structure $< F[x]|_\ell, +, \cdot >$ is a commutative ring. For special choices of the reduction polynomial $m(x)$, the structure becomes a field.

Definition 2.1.5. *A polynomial $d(x)$ is* irreducible *over the field* GF(p) *iff there exist no two polynomials $a(x)$ and $b(x)$ with coefficients in* GF(p) *such that $d(x) = a(x) \times b(x)$, where $a(x)$ and $b(x)$ are of degree > 0.*

The inverse element for the multiplication can be found by means of the extended Euclidean algorithm (see e.g. [68, p. 81]). Let $a(x)$ be the polynomial we want to find the inverse for. The extended Euclidean algorithm can then be used to find two polynomials $b(x)$ and $c(x)$ such that:

$$a(x) \times b(x) + m(x) \times c(x) = \gcd(a(x), m(x)). \tag{2.25}$$

Here $\gcd(a(x), m(x))$ denotes the greatest common divisor of the polynomials $a(x)$ and $m(x)$, which is always equal to **1** iff $m(x)$ is irreducible. Applying modular reduction to (2.25), we get:

$$a(x) \times b(x) \equiv \mathbf{1} \pmod{m(x)}, \tag{2.26}$$

which means that $b(x)$ is the inverse element of $a(x)$ for the definition of the multiplication '$\cdot$' given in (2.24).

Conclusion. Let F be the field GF(p). With a suitable choice for the reduction polynomial, the structure $< F[x]|_n, +, \cdot >$ is a field with p^n elements, usually denoted by GF(p^n).

2.1.6 Polynomials and Bytes

According to (2.22) a byte can be considered as a polynomial with coefficients in GF(2):

$$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 \mapsto b(x) \tag{2.27}$$

$$b(x) = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0. \tag{2.28}$$

The set of all possible byte values corresponds to the set of all polynomials with degree less than eight. Addition of bytes can be defined as addition of the corresponding polynomials. In order to define the multiplication, we need to select a reduction polynomial $m(x)$.

In the specification of Rijndael, we consider the bytes as polynomials. Byte addition is defined as addition of the corresponding polynomials. In order to define byte multiplication, we use the following irreducible polynomial as reduction polynomial:

$$m(x) = x^8 + x^4 + x^3 + x + 1. \tag{2.29}$$

Since this reduction polynomial is irreducible, we have constructed a representation for the field $\mathrm{GF}(2^8)$. Hence we can state the following: In the Rijndael specification, bytes are considered as elements of $\mathrm{GF}(2^8)$. Operations on bytes are defined as operations in $\mathrm{GF}(2^8)$.

Example 2.1.8. In our representation for $\mathrm{GF}(2^8)$, the product of the elements denoted by `57` and `83` is the element denoted by `C1`, since:

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1) &\times (x^7 + x + 1) \\ &= (x^{13} + x^{11} + x^9 + x^8 + x^7) \oplus (x^7 + x^5 + x^3 + x^2 + x) \\ &\quad \oplus (x^6 + x^4 + x^2 + x + 1) \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\end{aligned}$$

and

$$\begin{aligned}(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \\ \equiv x^7 + x^6 + 1 \pmod{x^8 + x^4 + x^3 + x + 1}.\end{aligned}$$

As opposed to addition, there is no simple equivalent processor instruction.

2.1.7 Polynomials and Columns

In the Rijndael specification, 4-byte columns are considered as polynomials over $\mathrm{GF}(2^8)$, having a degree smaller than four. In order to define the multiplication operation, the following reduction polynomial is used:

$$l(x) = x^4 + 1. \tag{2.30}$$

This polynomial is reducible, since in $\mathrm{GF}(2^8)$

$$x^4 + 1 = (x + 1)^4. \tag{2.31}$$

In the definition of Rijndael, one of the inputs of the multiplication is a constant polynomial.

Since $l(x)$ is reducible over $\mathrm{GF}(2^8)$, not all polynomials have an inverse element for the multiplication modulo $l(x)$. A polynomial $b(x)$ has an inverse if the polynomial $x + 1$ does not divide it.

Multiplication with a fixed polynomial. We work out in more detail the multiplication with the fixed polynomial used in Rijndael.

Let $b(x)$ be the fixed polynomial with degree three:

$$b(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 \tag{2.32}$$

and let $c(x)$ and $d(x)$ be two variable polynomials with coefficients c_i and d_i, respectively ($0 \leq i < 4$). We derive the matrix representation of the transformation that takes as input the coefficients of polynomial c, and produces as output the coefficients of the polynomial $d = b \times c$. We have:

$$\begin{aligned} & d = b \cdot c && (2.33) \\ & \Updownarrow \\ & (b_0 + b_1 x + b_2 x^2 + b_3 x^3) \times (c_0 + c_1 x + c_2 x^2 + c_3 x^3) \\ & \quad \equiv (d_0 + d_1 x + d_2 x^2 + d_3 x^3) \pmod{x^4 + 1} && (2.34) \end{aligned}$$

Working out the product and separating the conditions for different powers of x, we get:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} b_0 & b_3 & b_2 & b_1 \\ b_1 & b_0 & b_3 & b_2 \\ b_2 & b_1 & b_0 & b_3 \\ b_3 & b_2 & b_1 & b_0 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}. \tag{2.35}$$

2.2 Linear Codes

In this section we give a short introduction to the theory of linear codes. For a more detailed treatment, we refer the interested reader to the work of MacWilliams and Sloane [63]. In code theory textbooks, it is customary to write codewords as $1 \times n$ matrices, or *row vectors.* We will follow that custom here. In further chapters, one-dimensional arrays will as often be denoted as $n \times 1$ matrices, or *column vectors.*

2.2.1 Definitions

The *Hamming weight* of a codeword is defined as follows.

Definition 2.2.1. *The* Hamming weight $\mathrm{w_h}(\mathbf{x})$ *of a vector* $\mathbf{x}$ *is the number of nonzero components of the vector* $\mathbf{x}$.

Based on the definition of Hamming weight, we can define the *Hamming distance* between two vectors.

Definition 2.2.2. *The* Hamming distance *between two vectors* $\mathbf{x}$ *and* $\mathbf{y}$ *is* $\mathrm{w_h}(\mathbf{x}-\mathbf{y})$*, which is equal to the Hamming weight of the difference of the two vectors.*

Now we are ready to define linear codes.

Definition 2.2.3. *A* linear $[n,k,d]$ code *over* $\mathrm{GF}(2^p)$ *is a* k*-dimensional subspace of the vector space* $\mathrm{GF}(2^p)^n$*, where any two different vectors of the subspace have a Hamming distance of at least* d *(and* d *is the largest number with this property).*

The distance d of a linear code equals the minimum weight of a non-zero codeword in the code. A linear code can be described by each of the two following matrices:

1. A *generator* matrix G for an $[n,k,d]$ code $\mathcal{C}$ is a $k \times n$ matrix whose rows form a vector space basis for $\mathcal{C}$ (only generator matrices of full rank are considered). Since the choice of a basis in a vector space is not unique, a code has many different generator matrices that can be reduced to one another by performing elementary row operations. The *echelon form* of the generator matrix is the following:

$$\mathsf{G}_e = \left[\mathsf{I}_{k\times k}\ \mathsf{A}_{k\times(n-k)}\right], \tag{2.36}$$

   where $\mathsf{I}_{k\times k}$ is the $k \times k$ identity matrix.
2. A *parity-check* matrix H for an $[n,k,d]$ code $\mathcal{C}$ is an $(n-k) \times k$ matrix with the property that a vector $\mathbf{x}$ is a codeword of $\mathcal{C}$ iff

$$\mathsf{H}\mathbf{x}^{\mathrm{T}} = 0. \tag{2.37}$$

If G is a generator matrix and H a parity-check matrix of the same code, then

$$\mathsf{G}\mathsf{H}^{\mathrm{T}} = 0. \tag{2.38}$$

Moreover, if $\mathsf{G} = [\mathsf{I}\ \mathsf{C}]$ is a generator matrix of a code, then $\mathsf{H} = \left[-\mathsf{C}^{\mathrm{T}}\ \mathsf{I}\right]$ is a parity-check matrix of the same code.

The *dual code* $\mathcal{C}^{\perp}$ of a code $\mathcal{C}$ is defined as the set of vectors that are orthogonal to all the vectors of $\mathcal{C}$:

$$\mathcal{C}^{\perp} = \{\mathbf{x} \mid \mathbf{x}\mathbf{y}^{\mathrm{T}} = 0, \forall\ \mathbf{y} \in \mathcal{C}\}. \tag{2.39}$$

It follows that a parity-check matrix of $\mathcal{C}$ is a generator matrix of $\mathcal{C}^{\perp}$ and vice versa.

2.2.2 MDS codes

The theory of linear codes addresses the problems of determining the distance of a linear code and the construction of linear codes with a given distance. We review a few well-known results.

The Singleton bound gives an upper bound for the distance of a code with given dimensions.

Theorem 2.2.1 (The Singleton bound). *If $\mathcal{C}$ is an $[n, k, d]$ code, then $d \leq n - k + 1$.*

A code that meets the Singleton bound, is called a *maximal distance separable (MDS) code.* The following theorems relate the distance of a code to properties of the generator matrix G.

Theorem 2.2.2. *A linear code $\mathcal{C}$ has distance d iff every $d - 1$ columns of the parity check matrix H are linearly independent and there exists some set of d columns that are linearly dependent.*

By definition, an MDS-code has distance $n - k + 1$. Hence, every set of $n - k$ columns of the parity-check matrix are linearly independent. This property can be translated to a requirement for the matrix A:

Theorem 2.2.3 ([63]). *An $[n, k, d]$ code with generator matrix*

$$\mathsf{G} = \left[\mathsf{I}_{k\times k}\ \mathsf{A}_{k\times(n-k)}\right],$$

is an MDS code iff every square submatrix of A is nonsingular.

A well-known class of MDS codes is formed by the Reed-Solomon codes, for which efficient construction algorithms are known.

2.3 Boolean Functions

The smallest finite field has an order of 2: GF(2). Its two elements are denoted by **0** and **1**. Its addition is the integer addition modulo 2 and its multiplication is the integer multiplication modulo 2. Variables that range over GF(2) are called *Boolean variables*, or *bits* for short. The addition of 2 bits corresponds with the Boolean operation *exclusive or*, denoted by XOR. The multiplication of 2 bits corresponds to the Boolean operation AND. The operation of changing the value of a bit is called *complementation.*

A vector whose coordinates are bits is called a *Boolean vector.* The operation of changing the value of all bits of a Boolean vector is called complementation.