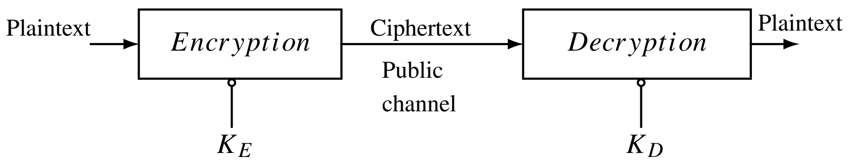
# 1

# Introduction to cryptography, codes, Boolean, and vectorial functions

## 1.1 Cryptography

A fundamental objective of *cryptography* is to enable two persons to communicate over an insecure channel (a public channel such as the internet) in such a way that any other person is unable to recover their messages (constituting the *plaintext*) from what is sent in its place over the channel (the *ciphertext*). The transformation of the plaintext into the ciphertext is called *encryption*, or enciphering. It is ensured by a *cryptosystem*. Encryption–decryption is the most ancient cryptographic activity (ciphers already existed in the fourth century BC) but its nature has deeply changed with the invention of computers, because the *cryptanalysis* (the activity of the third person, the eavesdropper, who aims at recovering the message, or better, the secret data used by the algorithm – which is assumed to be public) can use their power. Another important change will occur (see *e.g.*, [70, 360, 832]), at least for public-key cryptography (see the definition below), when quantum computers become operational.

The encryption algorithm takes as input the plaintext and an encryption key $K_E$, and it outputs the ciphertext. The *decryption* (or deciphering) algorithm takes as input the ciphertext and a private[1] decryption key $K_D$. It outputs the plaintext.



For being considered robust, a cryptosystem should not be cryptanalyzed by an attack needing less than $2^{80}$ elementary operations (which represent thousands of centuries of computation with a modern computer) and less than billions of plaintext–ciphertext pairs. In particular, an exhaustive search of the secret parameters of the cryptosystem (consisting in trying every possible value of them until the data given to the attacker match the computed data) should not be feasible in less than $2^{80}$ elementary operations. In fact, we most often even want that there is no faster cryptanalysis than exhaustive search.

[1] According to principles already stated in 1883 by A. Kerckhoffs [688], who cited a still more ancient manuscript by R. du Carlet [207], only the key(s) need absolutely to be kept secret – the confidentiality should not rely on the secrecy of the encryption method – and a cipher cannot be considered secure if it can be decrypted by the designer himself without using the decryption key.

Note that the term of cryptography is often used indifferently for naming the two activities of designing cryptosystems and of cryptanalyzing them, while the correct term when dealing with both is *cryptology*.

### 1.1.1 Symmetric versus public-key cryptosystems

If the encryption key is supposed to be secret, then we speak of *conventional cryptography* or of *private-key cryptography*. We also speak of *symmetric cryptography* since the same key can then be used for $K_E$ and $K_D$. In practice, the principle of conventional cryptography relies then on the sharing of a private key between the sender of a message (often called Alice) and its receiver (often called Bob). Until the late 1970s, only symmetric ciphers existed.

If the encryption key can be public, then we speak of *public-key cryptography* (or *asymmetric cryptography*), which is preferable to conventional cryptography, since it makes it possible to securely communicate without having previously shared keys in a secure way: every person who wants to receive secret messages can keep secret a decryption key and publish an encryption key; if $n$ persons want to secretly communicate pairwise using a public-key cryptosystem, they need $n$ encryption keys and $n$ decryption keys, when conventional cryptosystems will need $\binom{n}{2} = \frac{n(n-1)}{2}$ keys. Of course, it must be impossible to deduce in reasonable time, even with huge computational power, the private decryption key from the public encryption key. Such requirement is related to the problem of building *one-way function*s, that is, functions such that computing the image of an element is fast (*i.e.*, is a problem of polynomial complexity), while the problem of computing the preimage of an element has exponential complexity.

All known public-key cryptosystems, such as RSA, which uses operations in large rings [846], allow a much lower data throughput; they also need keys of sizes 10 times larger than symmetric ciphers for ensuring the same level of security. Some public-key cryptosystems, such as those of McEliece and Niederreiter (based on codes) [846], are faster, but have drawbacks, because the ciphertext and the plaintext have quite different lengths, and the keys are still larger than for other public-key cryptosystems.[2] Private-key cryptosystems are then still needed nowadays for ensuring the confidential transfer of large data. In practice, they are widely used for confidentiality in the internet, banking, mobile communications, etc., and their study and design are still an active domain of research. Thanks to public-key cryptosystems, the share-out of the necessary secret keys for the symmetric cipher can be done without using a secure channel (the secret keys for conventional cryptosystems are strings of a few hundreds of bits only and can then be encrypted by public-key cryptosystems). The protagonists can then exchange safely, over a public channel such as the internet, their common private encryption–decryption key, called a *session key*. Protocols specially devoted to key exchange can also be used.

The change caused by the intervention of quantum computers will be probably much less important for symmetric than for public-key cryptography. Most current symmetric ciphers

[2] Code-based, lattice-based, and other “postquantum” cryptosystems are, however, actively studied, mainly because they would be alternatives to RSA and to the cryptosystems based on the discrete logarithm, in case an efficient quantum computer could be built in the future, which would break them.

seem secure against attacks by quantum computers (Grover's algorithm [576], which, given a black box with $N$ possible inputs and some output, deduces with high probability from the results of $\mathcal{O}(\sqrt{N})$ evaluations the supposedly unique input,[3] will probably have as an impact the necessity to double the length of the keys).

### *1.1.2 Block ciphers versus stream ciphers*

The encryption in a symmetric cipher can be treated block by block in a so-called *block cipher* (such as the Advanced Encryption Standard, AES [403, 404]). The binary plaintext is then divided into blocks of the same size, several blocks being encrypted with the same key (and a public data called initial vector being changed more often). It can also be treated in a *stream cipher* [463], through the addition, most often mod 2, of a *keystream* of the same size as the plaintext, output by a pseudorandom generator (PRG) parameterized by a secret key (the keystream can be produced symbol by symbol, or block by block when the PRG uses a block cipher in a proper mode[4]). A quality of stream ciphers is to avoid error propagation, which gives them an advantage in applications where errors may occur during the transmission.

The ciphertext can be decrypted in the case of block ciphers by inverting the process and in the case of stream ciphers by the same bitwise addition of the keystream, which gives back the plaintext. Stream ciphers are also meant to be faster and to consume less electric power (which makes them adapted to cheap embedded devices). The triple constraint of being lightweight and fast while ensuring security is a difficult challenge for stream ciphers, all the more since they do not have the advantage of involving several rounds like block ciphers (their security is dependent on the PRG only). And the situation is nowadays still more difficult because modern block ciphers such as the AES are very fast. This difficulty has been illustrated by the failure of all six stream ciphers submitted to the 2000–2003 NESSIE project (New European Schemes for Signatures, Integrity and Encryption) [901], whose purpose was to identify secure cryptographic primitives. NESSIE has then been followed by the contest eSTREAM [495] organized later, between 2004 and 2008, by the European Union (EU) ECRYPT network.

As mentioned in [242], the price to pay for these three constraints described above is that security proofs hardly exist for efficient stream ciphers as they do for block ciphers. This is a drawback of stream ciphers, compared to block ciphers.[5] The only practical possibility for verifying the security of efficient stream ciphers (in particular, the unpredictability of the keystream they generate) is to prove that they resist the known attacks. It is then advisable to include some amount of randomness in them, so as to increase the probability of resisting future attacks.[6]

[3] Or equivalently finds with high probability a specific entry in an unsorted database of $N$ entries.
[4] Note, however, that stream ciphers are often supposed to be used on lighter devices than block ciphers (typically not needing cryptoprocessors, for instance).
[5] However, the security of block ciphers is actually proved under simplifying hypotheses, and it has been said by Lars Knudsen that "what is provably secure is probably not."
[6] Some stream cipher proposals, such as the Toyocrypt, LILI-128 and SFINKS ciphers, learned this at their own expense; see [387].

Proving the security of a cipher consists of reducing it to the intractability of a hard problem (a problem that has been extensively addressed by the academic community, and for which only algorithms of exponential or subexponential complexity could be found), implying that any potential attack on it could be used for designing an efficient algorithm (whose worst-case complexity would be polynomial in the size of its input) solving the hard problem.

Note that provably secure stream ciphers do exist (some proposals are even unconditionally secure, that is, are secure even if the attacker has unlimited computational power, but limited storage or access); see for instance the proposals by Alexi–Chor–Goldreich–Schnorr (whose security is reducible to the intractability of the RSA problem) or Blum–Blum–Shub [98] (whose security is reducible to the intractability of the quadratic residue problem modulo $pq$, where $p$ and $q$ are large primes), or the stream cipher QUAD [61] (based on the iteration of a multivariate quadratic system over a finite field, and whose security is reducible to the intractability of the so-called multivariate quadratic (MQ) polynomial problem). But they are too slow and too heavy for being used in practice. Even in the case of QUAD, which is the fastest, the encryption speed is lower than for the AES. And this is still worse when security is ensured unconditionally. This is why the stream ciphers using Boolean functions (see below) are still much used and studied.

## 1.2 Error-correcting codes

The objective of error-detecting/-correcting codes in *coding theory* is to enable digital communication over a noisy channel, in such a way that the errors of transmission can be detected by the receiver and, in the case of error correcting codes, corrected. General references are [63, 780, 809]. Shannon's paper [1033] is also prominent.

Without correction, when an error is detected, the information needs to be requested again by the receiver and sent again by the sender (such procedure is called an Automatic Repeat reQuest, ARQ). This is what happened with the first computers: working with binary words, they could detect only one error (one bit) in the transmission of $(x_1, \dots, x_k)$, by adding a parity bit $x_{k+1} = \bigoplus_{i=1}^{k} x_i$ (this transformed the word of length $k$ into a word of length $k+1$ having even *Hamming weight*, *i.e.*, an even number of nonzero coordinates, which was then sent over the noisy channel; if an error occurred in the transmission, then, assuming that only one could occur, this was detected by the fact that the received word had odd Hamming weight).

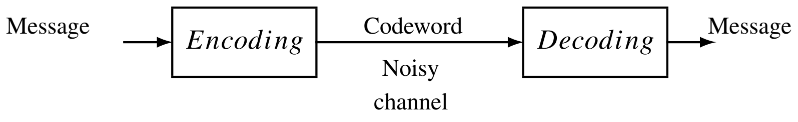
With correction, the ARQ is not necessary, but this requires in practice that fewer errors have occurred than for detection (see below). Hybrid coding techniques exist then that make a trade-off between the two approaches.

The aim of error detection/correction is achieved by using an encoding algorithm that transforms the information (assumed to be a sequence over some alphabet $\mathcal{A}$) before sending it over the channel. In the case of block coding,[7] the original sequence (the *message*) is treated as a list of vectors (words) of the same length – say $k$ – called *source vectors* which are encoded into *codeword*s of a larger length – say[8] $n$. If the alphabet with which the words

[7] We shall not address convolutional coding here.
[8] When dealing with Boolean functions, the symbol $n$ will be often devoted to their number of variables; the length of the codes they will constitute will then not be $n$ but $N = 2^n$. See Section 1.3.

are built is the field $\mathbb{F}_2$ of order 2, we say that the code is binary. If the code is not binary, then the symbols of the alphabet will have to be transformed into binary vectors before being sent over a binary channel.

Thanks to the length extension, called *redundancy*, the codewords sent over the channel are some of all possible vectors of length $n$. The set $C$ of all codewords is called the *code* (for instance, in the case of the detecting codes using a parity bit as indicated above, the code is made of all binary words of length $n = k + 1$ and of even Hamming weights; it is called the *parity code*). The only information the receiver has, concerning the sent word, is that it belongs to $C$.



### 1.2.1 Detecting and correcting capacities of a code

The decoding algorithm of an error-detecting code is able to recognize if a received vector is a codeword. This makes possible to detect errors of transmission if (see [585]) denoting by $d$ the minimum *Hamming distance* between codewords, *i.e.*, the minimum number of positions at which codewords differ (called the *minimum distance* of the code), no more than $d - 1$ coordinates of the received vector differ from those of the sent codeword (condition for having no risk that a codeword different from the sent one can be received and then accepted). In the case of an error-correcting code, the decoding algorithm can additionally correct the errors of transmission, if their number is smaller than or equal to the so-called *correction capacity* of the code. This capacity equals $e = \left\lfloor \frac{d-1}{2} \right\rfloor$, where "$\lfloor\ \rfloor$" denotes the integer part (and so, roughly, a code can detect twice as many errors than it can correct), since the condition for having no risk that a vector corresponds, as received vector, to more than one sent codeword with at most $t$ errors of transmission in each case is that $2t < d$. Indeed, in order to be always able (theoretically) to recover the correct codeword, we need that, for every word $y$ at distance at most $t$ from a codeword $x$, there does not exist another codeword $x'$ at distance at most $t$ from $y$, and this is equivalent to saying that the Hamming distance between any two different codewords is larger than or equal to $2t + 1$:

- If there exist a vector $y$ and two codewords $x$ and $x'$ at Hamming distance at most $t$ from $y$, then we have $d \leq 2t$ by the triangular inequality on distances.
- Conversely, if there exist two codewords $x$ and $x'$ at Hamming distance $\delta \leq 2t$ from each other, then there exists a vector $y$ such that $d_H(x, y) \leq t$ and $d_H(x', y) \leq t$ (let $I$ be the set of positions where $x$ and $x'$ coincide; take $y_i = x_i$ when $i \in I$ and among the $\delta$ others, take for instance $\lfloor \frac{\delta}{2} \rfloor$ coordinates of $y$ equal to those of $x$ and the $\lceil \frac{\delta}{2} \rceil$ others equal to those of $x'$).

In practice, determining $d$ and then $e = \left\lfloor \frac{d-1}{2} \right\rfloor$ and showing that they are large is not sufficient. We still need to have an efficient decoding algorithm to recover the sent codeword. The naive method consisting in visiting all codewords and keeping the nearest one from the received word is inefficient because the number $2^k$ of codewords is too large, in general.

Determining the nearest codeword from a received vector is called *maximum likelihood decoding*.

The correction capacity $e$ is limited by the *Hamming bound* (or *sphere-packing bound*): since all the balls $B(x,e) = \{y \in \mathcal{A}^n; d_H(x,y) \leq e\}$, of radius $e$ and centered in codewords are pairwise disjoint, and since there are $|C|$ of them, the size of their union equals $|C| \sum_{i=0}^{e} \binom{n}{i}(q-1)^i$, where $q$ is the size of the alphabet. This union is a subset of $\mathcal{A}^n$. This implies the following:

$$|C| \sum_{i=0}^{e} \binom{n}{i}(q-1)^i \leq q^n.$$

The codes that achieve this bound with equality are called *perfect code*s.

*Puncturing, shortening, and extending codes*

The *punctured code* of a code $C$ is the set of vectors obtained by deleting the coordinate at some fixed position $i$ in each codeword of $C$; we shall call such transformation *puncturing at position i*. This operation can be iterated, and we shall still speak of puncturing a code when deleting the codeword coordinates at several positions.

The *shortened code* of a code $C$ is the set of vectors obtained by keeping only those codewords whose $i$th coordinate is null and deleting this $i$th coordinate.

The *extended code* of a code $C$ over an additive group is the set of vectors, say, $(c_0, c_1, \dots, c_n)$, where $(c_1, \dots, c_n) \in C$ and $c_0 = -(c_1 + \cdots + c_n)$. Note that the extended code of $C$ equals the intersection of the code $\{(c_0, c_1, \dots, c_n) \in \mathbb{F}_q;\ (c_1, \dots, c_n) \in C\}$ and of the parity code $(c_0, c_1, \dots, c_n) \in \mathbb{F}_q;\ \sum_{i=0}^{n} x_i = 0\}$.

### 1.2.2 Parameters of a code

Sending words of length $n$ over the channel instead of words of length $k$ slows down the transmission of information in the ratio of $\frac{k}{n}$. This ratio, called the *transmission rate*, must be as high as possible, for a given correction capacity, to make possible fast communication. As we see, the three important parameters of a code $C$ are $n, k, d$ (or equivalently $n, |C|, d$ since if $q$ is the alphabet's size, we have $|C| = q^k$), and the first aim[9] of algebraic coding is to find codes minimizing $n$, maximizing $k$, and maximizing $d$, for diverse ranges of parameters corresponding to the needs of communication (see tables of best-known codes in [570]). It is easily seen that $k \leq n-d+1$ (this inequality, valid for any code over any alphabet, is called the *Singleton bound*) since erasing the coordinates of all codewords at $d-1$ fixed positions gives a set of $q^k$ distinct vectors of length $n-d+1$, where $q$ is the size of the alphabet, and the number of all vectors of length $n-d+1$ equals $q^{n-d+1}$. Codes achieving the Singleton bound with equality are called *maximum distance separable* (MDS). In the case of binary linear codes (see below), it can be shown by using the Pless identities (see, *e.g.*, [348]) that MDS codes have dimension at most 1 or at least $n-1$ and, except for such codes, the bound becomes then $k \leq n-d$.

Another important parameter is the *covering radius*, which is the smallest integer $\rho$ such that the spheres of (Hamming) radius $\rho$ centered at the codewords cover the whole space. In

[9] The second aim is to find decoding algorithms for the codes found.

other words, it is the minimal integer $\rho$ such that every vector of length $n$ lies at Hamming distance at most $\rho$ from at least one codeword, that is, the maximum number of errors to be corrected when maximum likelihood decoding (see page 6) is used. The book [375] is devoted to its study.

The sphere-covering bound is the lower bound on the covering radius $\rho$, which expresses that, by definition, the balls $B(x, \rho) = \{y \in \mathcal{A}^n; d_H(x, y) \leq \rho\}$, of radius $\rho$ and centered in codewords, cover the whole space $\mathcal{A}^n$:

$$|C| \sum_{i=0}^{\rho} \binom{n}{i} (q-1)^i \geq q^n.$$

### *1.2.3 Linear codes*

The general class of linear codes gives a simple and wide example of codes and how they can be used in error correction.

**Definition 1** *A code is called a* linear code *if its alphabet is a finite field $\mathbb{F}_q$ (where $q$ is the power of a prime) and if it has the structure of an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^n$, where $n$ is its length (see [809]).*

A code that is not necessarily linear is called an *unrestricted code*. The minimum distance of a linear code equals the minimum Hamming weight of all nonzero codewords, since the Hamming distance between two vectors equals the Hamming weight of their difference. We shall write that a linear code[10] over $\mathbb{F}_q$ is an $[n, k, d]_q$*-code* (and if the value of $q$ is clear from the context, an $[n, k, d]$*-code*) if it has length $n$, dimension $k$, and minimum distance $d$. The translates of a linear code are called its *coset*s and the elements of minimum Hamming weights in these cosets are called *coset leader*s (there may exist several in some cosets).

#### *Generator matrix*

Any linear code can be described by a *generator matrix* $G$, obtained by choosing a basis of this vector space and writing its elements as the rows of this matrix. The code equals the set of all the vectors of the form $u \times G$, where $u$ ranges over $\mathbb{F}_q^k$ (and $\times$ is the matrix product) and a possible encoding algorithm is therefore the mapping $u \in \mathbb{F}_q^k \mapsto u \times G \in \mathbb{F}_q^n$. When the codeword corresponding to a given source vector $u$ is obtained by inserting so-called *parity check coordinates* in the source vector (whose coordinates are then called *information coordinates*), the code is called *systematic* (it equals then the *graph* $\{(x, F(x), x \in \mathbb{F}_q^k\}$ of a function, up to coordinate permutation). The corresponding generator matrix is then called a *systematic generator matrix* and has the form $[I_k : M]$, where $I_k$ is the $k \times k$ identity matrix, up to column permutation. It is easily seen that every linear code has such a generator matrix: any generator matrix (of rank $k$) has $k$ linearly independent columns, and if we place these columns at the $k$ first positions, we obtain $G = [A : M]$, where $A$ is a nonsingular $k \times k$ matrix; then $A^{-1} \times G = [I_k : A^{-1} \times M]$ is a systematic generator matrix of the permuted

[10] The square brackets around $n, k, d$ specify that the code is linear, contrary to standard parentheses.

code (since the multiplication by $A^{-1}$ transforms a basis of the permuted code into another basis of the permuted code).

### *Dual code and parity check matrix*

The generator matrix is well suited for generating the codewords, but it is not for checking if a received word of length $n$ is a codeword or not. A characterization of the codewords is obtained thanks to the generator matrix $H$ of the *dual code* $C^\perp = \{x \in \mathbb{F}_q^n;\ \forall y \in C,\ x \cdot y = \sum_{i=1}^n x_i\, y_i = 0\}$ (such a matrix is called a *parity check matrix* and "$\cdot$" is called the *usual inner product*, or scalar product, in $\mathbb{F}_q^n$): we have $x \in C$ if and only if $x \times H^t$ is the null vector. Consequently, the minimum distance of any linear code equals the minimum number of $\mathbb{F}_q$-linearly dependent columns in one of its parity check matrices (any one). For instance, the binary *Hamming code* of length $n = 2^m - 1$, which has by definition for parity check matrix the $m \times (2^m - 1)$ binary matrix whose columns are all the nonzero vectors of $\mathbb{F}_2^m$ in some order, has minimum distance 3. This code, which by definition is unique up to equivalence, has played an important historical role since it is the first perfect code found. It still plays a role since many computers use it to detect errors in their internal communications. It is the basis on which BCH and Reed–Muller codes were built (see pages 10 and 151). It depends on the choice of the order, but we say that two codes over $\mathbb{F}_q$ are *equivalent codes* if they are equal, up to some permutation of the coordinates of their codewords (and, for nonbinary codes, to the multiplication of each coordinate in each codeword by a nonzero element of $\mathbb{F}_q$ depending only on the position of this coordinate). Note that such codes have the same parameters.

The dual of the binary Hamming code is called the *simplex code*. A generator matrix of this code being the parity check matrix of the Hamming code described above, and the rows of this matrix representing then the *coordinate functions* in $\mathbb{F}_2^m$ (sometimes called dictator functions), on which the order chosen for listing the values is given by the columns of the matrix, the codewords of the simplex code are the lists of values taken on $\mathbb{F}_2^m \setminus \{0_m\}$ by all linear functions.

Note that the dual of a linear code $C$ permuted by some bijection over the indices equals $C^\perp$ permuted by this same bijection, and that, if $G = [I_k : M]$ is a systematic generator matrix of a linear code $C$, then $[-M^t : I_{n-k}]$ is a parity check matrix of $C$, where $M^t$ is the transposed matrix of $M$.

The linear codes that are supplementary with their duals (or equivalently that have trivial intersection with their duals since the dimensions of a code and of its dual are complementary to $n$) are called complementary dual codes (LCD) and will play an important role in Subsection 12.1.5.

### *The advantages of linearity*

Linearity allows considerably simplifying some main issues about codes. Firstly, the minimum distance being equal to the minimum nonzero Hamming weight, computing it (if it cannot be determined mathematically) needs only to visit $q^k - 1$ codewords instead of $\frac{q^k(q^k-1)}{2}$ pairs of codewords. Secondly, the knowledge of the code is provided by a $k \times n$ generator matrix and needs then the description of $k$ codewords instead of all $q^k$ codewords.

Thirdly, a general decoding algorithm is valid for every linear code. This algorithm is not efficient in general, but it gives a framework for the efficient decoding algorithms that will have to be found for each class of linear codes. The principle of this algorithm is as follows: let $y$ be the (known) received vector corresponding to the (unknown) sent codeword $x$. We assume that there has been at most $d-1$ errors of transmission, where $d$ is the minimum distance, if the code is used for error detection, and at most $e$ errors of transmission, where $e = \lfloor (d-1)/2 \rfloor$ is the correcting capacity of the code, if the code is used for error correction. The error detection is made by checking if the so-called *syndrome* $s = y \times H^t$ is the zero vector. If it is not, then denoting by $\epsilon$ the so-called (unknown) *error vector* $\epsilon = y - x$, correcting the errors of transmission is equivalent to determining $\epsilon$. This can be done by visiting all vectors $z$ of Hamming weight at most $e$ in $\mathbb{F}_q^n$ and checking if $z \times H^t = s$ (indeed, by linearity of matrix multiplication, the syndrome of the error vector equals the syndrome of the received vector, which is known). There exists a unique $z$ of Hamming weight at most $e$ in $\mathbb{F}_q^n$ such that $z \times H^t = s$; this unique $z$ equals $\epsilon$.

## *Concatenating codes*

Given an $\mathbb{F}_q$-linear $[n,k,d]$ code $C$ (where $n$ is the length, $k$ is the dimension, and $d$ is the minimum distance), where $q = 2^e$, $e \geq 2$, a binary $[n',e,d']$ code $C'$ and an $\mathbb{F}_2$-isomorphism $\phi : \mathbb{F}_q \mapsto C'$, the *concatenated code* $C''$ equals the $[nn', ke, d'' \geq dd']$ binary code $\{(\phi(c_1), \dots, \phi(c_n)); (c_1, \dots, c_n) \in C\}$. Codes $C$ and $C'$ are respectively called outer code and inner code for this construction.

## *MDS linear codes*

Let $C$ be an $[n,k,d]$ code over a field $K$, let $H$ be its parity check matrix, and $G$ its generator matrix. Then $n-k$ is the rank of $H$, and we have then $d \leq n-k+1$ since $n-k+1$ columns of $H$ are always linearly dependent and therefore any set of indices of size $n-k+1$ contains the support of a nonzero codeword. This proves again the Singleton bound: $d \leq n-k+1$.

Recall that $C$ is called *MDS* if $d = n-k+1$. The following are the properties of MDS linear codes:

1. $C$ is MDS if and only if each set of $n-k$ columns of $H$ has rank $n-k$.
2. If $C$ is MDS, then $C^\perp$ is MDS.
3. $C$ is MDS if and only if each set of $k$ columns of $G$ has rank $k$ (and their positions constitute then an information set; see page 161).

## *Other properties of linear codes*

Puncturing, shortening, and extending codes preserve their linearity. Puncturing preserves the MDS property (if $n > k$).

The following lemma will be useful when dealing with Reed–Muller codes in Chapter 4.

**Lemma 1** *Let $C$ be a linear code of length $n$ over $\mathbb{F}_q$ and $\hat{C}$ its extended code. We have* $\hat{C}^\perp = \{(y_0, \dots, y_n) \in \mathbb{F}_q^{n+1}; (y_1 - y_0, \dots, y_n - y_0) \in C^\perp\}$.

*Proof* We have $\hat{C}^\perp = \{(y_0, \dots, y_n) \in \mathbb{F}_q^{n+1}; \forall (x_1, \dots, x_n) \in C, y_0(-\sum_{i=1}^n x_i) + \sum_{i=1}^n x_i y_i = 0\} = \{(y_0, \dots, y_n) \in \mathbb{F}_q^{n+1}; \forall (x_1, \dots, x_n) \in C, \sum_{i=1}^n x_i(y_i - y_0) = 0\} = \{(y_0, \dots, y_n) \in \mathbb{F}_q^{n+1}; (y_1 - y_0, \dots, y_n - y_0) \in C^\perp\}$. □

**Uniformly packed codes:** These codes will play a role with respect to almost perfect nonlinear (APN) functions, at page 381.

**Definition 2** *[50] Let $C$ be any binary code of length $N$, with minimum distance $d = 2e+1$ and covering radius $\rho$. For any $x \in \mathbb{F}_2^N$, let us denote by $\zeta_j(x)$ the number of codewords of $C$ at distance $j$ from $x$. The code $C$ is called a* uniformly packed code, *if there exist real numbers $h_0, h_1, \dots, h_\rho$ such that, for any $x \in \mathbb{F}_2^N$, the following equality holds:*

$$\sum_{j=0}^{\rho} h_j \, \zeta_j(x) = 1.$$

As shown in [51], this is equivalent to saying that the covering radius of the code equals its external distance (*i.e.*, the number of different nonzero distances between the codewords of its dual).

### *1.2.4 Cyclic codes*

*Two-error correcting Bose–Chaudhuri–Hocquenghem (BCH) codes*

The binary Hamming code of length $n = 2^m - 1$ has dimension $n - m$ and needs $m$ parity check bits for being able to correct 1 error. It happens that 2-error binary correcting codes can be built with $2m$ parity check bits. Let us denote by $W_1, \dots, W_n$ the nonzero binary vectors of length $m$ written as columns in some order. The parity check matrix of the Hamming code of length $n = 2^m - 1$ is as follows:

$$H = [W_1, \dots, W_n].$$

To find a 2-error correcting code $C$ of the same length, we consider the codes whose parity check matrices $H'$ are the $2m \times n$ matrices whose $m$ first rows are those of $H$. These codes being subcodes of the binary Hamming code, they are at least 1-error correcting. For each such matrix $H'$, there exists a function $F$ from $\mathbb{F}_2^m$ to itself such that:

$$H' = \begin{bmatrix} W_1 & W_2 & \dots & W_n \\ F(W_1) & F(W_2) & \dots & F(W_n) \end{bmatrix}.$$

Note that, when $F$ is a *permutation* (*i.e.*, is bijective), the code of generator matrix $H'$ is a so-called *double simplex code* (and plays a central role in [136]); it is the direct sum of two simplex codes: the standard one and its permutation by $F$.

Going back to general $F$, assume that two errors are made in the transmission of a codeword of $C$, at indices $i \neq j$. The *syndrome* of the received vector equals that of the error vector, that is,

$$\begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} W_i \\ F(W_i) \end{bmatrix} + \begin{bmatrix} W_j \\ F(W_j) \end{bmatrix},$$