CS 880: Pseudorandomness and Derandomization 1/30/2013

Lecture 3: Error Correcting Codes

Instructors: Holger Dell and Dieter van Melkebeek Scribe: Xi Wu

In this lecture we review some background on error correcting codes (ECCs), in particular the Hadamard code (or Walsh–Hadamard code) and the Reed–Solomon code. Our motivation for doing so is two-fold:

- The theory of ECCs provides an interesting framework in which PRGs for certain classes of algorithms can be interpreted. We will see some examples of that in the next few lectures.
- The specific two ECCs above turn out to be useful tools for constructing PRGs for broad classes of algorithms.

In the latter context, good ECCs can be viewed as examples of successful derandomizations. This is because *random* ECCs often have good error correction properties, and the above codes achieve some of them explicitly. As such, it is not surprising that the codes are useful in other derandomization contexts. Of particular importance to us is the notion of *list decoding* of ECCs, which we will discuss in a later lecture.

1 General Codes

The original motivation for ECCs is to enable efficient and reliable message transmission over a noisy channel – be it memory storage or an actual transmission line. The message is encoded by adding some redundant information such that, if a small part of the encoded message gets corrupted, it can be corrected and the actual message be recovered.

ECCs can broadly be classified into two groups:

- Variable-length codes, in which messages of the same length can have encodings of different length. Such codes are useful in contexts where there is some prior knowledge about the distribution of the messages that need to be sent – more frequent messages would get shorter encodings.
- Block codes, in which the length of the encoding only depends on the length of the message. This is the type of ECCs that we will use.

Definition 1 (Code). *An (n, k)-code over the alphabet Σ is an injective mapping* $\mathrm{Enc} : \Sigma^k \mapsto \Sigma^n$.

We refer to an input $x \in \Sigma^k$ for Enc as a *message* and to k as the *message length* of the code. An image $y \in \mathrm{Enc}(\Sigma^k)$ is a *codeword* and n is the *block length*. Sometimes we use the term "code" also to refer to the range $C = \mathrm{Enc}(\Sigma^k)$ of the mapping, i.e., to the set of all codewords. We are mostly interested in codes over the binary alphabet $\Sigma = \{0, 1\}$, but we also consider codes over larger alphabets. If we want to make the size q of the alphabet explicit, we include it as a subscript in the code annotation and refer to an $(n, k)_q$-code.

Note that we do not require that $\mathrm{Enc}(x)$ contains the message x, i.e., we consider more general codes than those that "add redundant information". Codes that do contain the original message in the encoding are called *systematic*.

We do require the mapping in Definition 1 to be injective in order for the original message x to be retrievable from its encoding $y = \mathrm{Enc}(x)$. In fact, we would like this to be possible even if the codeword y gets corrupted in a few positions, that is, we can retrieve x from any string z such that $d_{\mathrm{H}}(y, z)$ is small, where

$$d_{\mathrm{H}}(y, z) \doteq \Big|\{i \in [n] \,:\, y_i \neq z_i\}\Big|$$

denotes the *Hamming distance* between y and z. A position $i \in [n]$ in which a retrieved word z differs from the encoding $y = \mathrm{Enc}(x)$ is called an *error*. How many errors can be tolerated is governed by the minimum Hamming distance between any two distinct codewords. This parameter is called the *distance* of the code. Since it only depends on the set C of codewords, we use the following notation.

Definition 2 (Distance of a code).

$$d_{\mathrm{H}}(C) \doteq \min_{y \neq y' \in C} d_{\mathrm{H}}(y, y') = \min_{x \neq x' \in \Sigma^k} d_{\mathrm{H}}(\mathrm{Enc}(x), \mathrm{Enc}(x')).$$

Consider a worst-case model of a noisy channel, in which an adversary can corrupt any but at most t positions of the codeword $y = \mathrm{Enc}(x)$ to produce the received word z. We say that the code can *detect* t errors if in this model we can always tell from the received word z whether any position in y has been corrupted or not. We say that the code can *correct* t errors if, in this model, we can always uniquely recover x from the received word z.

Proposition 1.

(a) C can detect t errors if and only if $d_{\mathrm{H}}(C) \geq t + 1$.

(b) C can correct t errors if and only if $d_{\mathrm{H}}(C) \geq 2t + 1$.

Proof. Part (a). C can detect t errors if and only if there is no way to jump from one codeword y to another codeword y' by changing at most t positions, which is the same as saying that $d_{\mathrm{H}}(C) > t$. Part (b). C can correct t errors if and only if there is no way to reach the same received word z from two distinct codewords y and y' by changing at most t positions in each. If there is a way to do this, then $d_{\mathrm{H}}(y, y') \leq d_{\mathrm{H}}(y, z) + d_{\mathrm{H}}(z, y') \leq 2t$, so $d_{\mathrm{H}}(C) \leq 2t$. Conversely, if $d_{\mathrm{H}}(C) \leq 2t$ then there are two distinct codewords y and y' such that $d_{\mathrm{H}}(y, y') \leq 2t$; in that case $d_{\mathrm{H}}(y, z) \leq t$ and $d_{\mathrm{H}}(z, y') \leq t$ holds for the string z obtained from y by changing $\lceil d_{\mathrm{H}}(y, y') \rceil$ of the positions in which y and y' differ such that they match y'. □

The factor 2 in part (b) of Proposition 1 implies that unique decoding can never go beyond half the distance of the code. Given the importance of the distance parameter, it is often included in the parameters describing the code; we denote a code $\mathrm{Enc} : \Sigma^k \mapsto \Sigma^n$ over an alphabet Σ with q elements and with distance at least d as an (n, k, d)-code or on $(n, k, d)_q$-code.

Two of the goals in code design are the following:

- Maximize the *relative distance* d/n so that the fraction of errors that can be corrected is as large as possible.

- Maximize the *rate* k/n so that the amount of redundancy is as small as possible.

These two goals are contending against each other. For example, high relative distance can easily be achieved at the cost of low rate by simply repeating the message many times. In the extreme, binary codes with relative distance 1 can contain no more than 2 codewords so the rate is no more than $1/n$. At the other extreme, the only codes with rate 1 are permutations, for which $d = 1$ and thus the relative distance is $1/n$. The study of the trade-off between the rate and the relative distance constitutes a large part of coding theory.

If we choose a mapping from Σ^k to Σ^n uniformly at random, we get a code that, with high probability, has a good relative distance. However, we need explicit codes, that is, codes that can be constructed efficiently without randomness. Apart from the relative distance and the rate, we care about the computational complexity of the encoding and decoding procedures. The best known explicit and efficient codes all belong to a class known as *linear codes*, which we discuss next.

2 Linear Codes

In the context of linear codes, we interpret the alphabet Σ as a finite field $\mathbb{F}_q$, and we restrict our attention to encoding functions Enc that are linear mappings. In coding theory, messages and codewords are usually represented as row vectors. We will go with the convention common in linear algebra and use column vectors instead.

Definition 3 (Linear Code). $\mathrm{Enc} : \mathbb{F}_q^k \mapsto \mathbb{F}_q^n$ *is a* linear code *if there exists a matrix* $G \in \mathbb{F}_q^{n \times k}$ *of full rank such that* $\mathrm{Enc}(x) = Gx$. *The matrix* G *is called the* generator matrix *of* Enc.

Note that the requirement of G having full rank is equivalent to Enc being injective. Linear codes have a much more succinct description than general ones – it suffices to describe the generator matrix G, i.e., kn elements, as opposed to the $2^k n$ elements needed to describe a general code. Moreover, the encoding boils down to a matrix-vector multiplication and is therefore efficient (assuming the generator matrix is efficiently computable). The decoding problem for linear codes is NP-hard in general, but there are efficient decoding algorithms for the specific linear codes that we consider in the next section. In the (n, k)-notation and its variants, we indicate the fact that the code is linear by using square brackets instead of normal brackets: $[n, k]$ or $[n, k, d]_q$.

In terms of the set C of codewords, the restriction to linear codes means that we only consider linear subspaces of $\mathbb{F}_q^n$. The subspace consists of all linear combinations of the columns of G, and therefore, its dimension is k. Once we have a vector space C, we can consider its dual, denoted $C^\perp$, which consists of all vectors that are orthogonal to C:

$$C^\perp \doteq \{y' \in \mathbb{F}_q^n \,:\, (\forall y \in C)\, \langle y, y' \rangle = 0\}\,,$$

where $\langle y, y' \rangle$ is the inner product of y and y',

$$\langle y, y' \rangle \doteq \sum_{i=1}^{n} y_i y'_i\,,$$

and the arithmetic is carried out over $\mathbb{F}_q$. The dual $C^\perp$ is itself a vector space and has dimension $n - k$. This means that it is the range of some linear mapping $\mathrm{Enc}^\perp : \mathbb{F}_q^{n-k} \to \mathbb{F}_q^n$ with $\mathrm{Enc}^\perp(x') = H^T x'$ for $x' \in \mathbb{F}_q^{n-k}$, where H^T is a matrix of full rank with n rows and $n - k$ columns,

the transpose of which is denoted by H. We call $\mathrm{Enc}^\perp$ a[1] *dual code* of Enc. In notation, the dual of an $[n,k]$-code is an $[n,n-k]$-code. The dual space of $C^\perp$ is C again, which means that taking the dual of the dual code gives back the original code.

The matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ is called a *parity check* matrix of Enc for the following reason.

Proposition 2. *$z \in \mathbb{F}_q^n$ is a codeword of* Enc *if and only if $Hz = 0$.*

Proof. Since C is the dual of $C^\perp$, we have that

$$\begin{aligned} z \in C &\iff (\forall y' \in C^\perp)\, \langle y', z\rangle = 0 \\ &\iff (\forall x' \in \mathbb{F}_q^{n-k})\, \langle H^T x', z\rangle = 0 \\ &\iff (\forall x' \in \mathbb{F}_q^{n-k})\, \langle x', Hz\rangle = 0 \\ &\iff Hz = 0. \end{aligned}$$

□

Note that in the binary case ($q = 2$), the condition $Hz = 0$ consists of $n-k$ parity checks for z. Hence the name "parity check matrix" for H. For a received word z, the entries of the $(n-k)$-dimensional vector Hz are called the *syndromes* of z. By Proposition 2, a received word is a codeword if and only if all of its syndromes vanish.

The distance $d_{\mathrm{H}}(C)$ of a linear code C with parity check matrix H can be characterized by the following proposition.

Proposition 3.

(a) $d_{\mathrm{H}}(C)$ equals the minimum Hamming weight *of a nonzero codeword, where the Hamming weight is the number of nonzero components.*

(b) $d_{\mathrm{H}}(C)$ equals the minimum number of columns in H that are linearly dependent.

Proof. Part (a) follows because $d_{\mathrm{H}}(y, y') = d_{\mathrm{H}}(y - y', 0)$ is the Hamming weight of $y - y'$, the fact that 0 is a codeword, and the fact that, if y and y' are codewords, then so is $y - y'$.
Part (b). By Proposition 2, z is a codeword if and only if $Hz = 0$. Thus, the minimum number of columns in H that are linearly dependent equals the minimum Hamming weight of a codeword, which equals $d_{\mathrm{H}}(C)$ by part (a). □

Note that in the binary case, a linear combination is just a sum over $\mathbb{F}_2^n$. Thus, the distance of a binary linear code is the smallest positive number of columns that sum to zero.

3 Specific Codes

We now discuss some specific linear codes and their decoding procedures.

[1] When viewed as a set of codewords, the dual of a linear code is uniquely defined, but not when viewed as a linear mapping.

3.1 Hamming

Hamming codes are a family of binary linear codes ($q = 2$) that we define in terms of their parity check matrices H. For each positive integer ℓ, we consider the matrix H whose columns are all nonzero vectors of length ℓ in lexicographical order. This matrix has full rank, and the smallest number of linearly dependent columns is 3. Hence, we have a $[2^\ell - 1, 2^\ell - \ell - 1, 3]_2$-code.

The code can detect two errors and can correct a single error. Single error correction is particularly simple because the syndrome not only tells us whether an error occurred or not (the syndrome is zero if and only if there is no error), but in case of an error it equals the index of the location at which the error occurred. Indeed, suppose that y is a codeword and the received word z has an error in position i: $z = y + e_i$, where e_i denotes the vector that has a 1 in position i and vanishes elsewhere. Then the syndrome equals

$$Hz = H(y + e_i) = Hy + He_i = 0 + i = i\,,$$

where i denotes the i-th column of H, that is, it is the binary representation of the integer i.

The rate of the Hamming code is very good. Combined with the simple procedure for single error correction, this makes the code very suitable for situations where errors are very rare, for example in memory chips. In complexity theory the Hamming code is not that useful because it only corrects one error.

3.2 Hadamard

In contrast, the Hadamard code is very useful in theory, but not in practice. This is because its relative distance is very high but its rate very low (which can be shown to be necessary for such high relative distance). The Hadamard code is also a binary linear code. It encodes a string x of length k as the inner product of x with all possible vectors of length k:

$$\mathrm{Enc}(x) = (\langle x, a\rangle)_{a\in\{0,1\}^k}.$$

Alternately, we can view $\mathrm{Enc}(x)$ as the value of every linear function

$$L_a : \mathbb{F}_2^k \mapsto \mathbb{F}_2 \quad : \quad y \mapsto \sum_{i=1}^{k} a_i y_i$$

at x, or as the value of the linear function L_x at all points a, i.e., as the function table of L_x.

Note that, for any two distinct $x, x' \in \{0,1\}^k$, the functions L_x and $L_{x'}$ agree in exactly half the points $a \in \{0,1\}^k$. Indeed, assuming without loss of generality that $x_1 \neq x'_1$, we have that

$$L_x(a) = L_{x'}(a) \iff a_1 = a_2(x_2 - x'_2) + \cdots + a_k(x_k - x'_k)\,,$$

so for any fixed $a_2, \ldots, a_k$, there is exactly one a_1 for which the equality holds. Therefore, there are 2^{k-1} points a on which L_x and $L_{x'}$ agree, and the same number of points on which they disagree. This means that the distance of the Hadamard code is 2^{k-1}.

In conclusion, the Hadamard code is a $[2^k, k, 2^{k-1}]_2$-code. Its relative distance of $1/2$ can be shown asymptotically optimal for binary codes (see for example, Chapter 4 of [Rot06]). However, its rate is exponentially small.

In fact, one component of the encoding contains no information whatsoever, namely the one for $a = 0$. Leaving out this component results in the *punctured Hadamard code*, which is a $[2^k - 1, k]_2$-code. Its generator matrix is H^T, where H denotes the parity check matrix of the $[2^k - 1, 2^k - k - 1]$-Hamming code. Thus, the punctured Hadamard code and the Hamming code are duals.

Local decoding. Given the relative distance of $1/2$, we can in principle recover the message $x \in \{0,1\}^k$ from any received word $z \in \{0,1\}^{2^k}$ in which less than a fraction $1/4$ of the positions differ from the correct encoding $y = \mathrm{Enc}(x)$. We would like to be able to do so in time polynomial in k. Deterministically, this is impossible for the same reason as in the example from Lecture 1 of computing the average of a function given as a black-box: A deterministic algorithm that runs in time polynomial in k can only look at a negligible portion of z, and we can tweak the rest of z such that the correct answer changes while the output of the algorithm remains the same. However, using randomness, we can efficiently recover any bit of x by only looking at a small number of positions of z. This is called *local decoding*.

In order to formalize the notion, we need to consider decoding algorithms Dec that have oracle access to the received word z. This means that Dec can use a procedure that takes as input an index a and returns the ath position of z in constant time. We refer to such an algorithm as an oracle algorithm, and write the oracle as a superscript to the algorithm: Dec^z.

Definition 4 (Local Decoder). *A local decoder up to relative error ϵ for an (n,k)-code* Enc *over Σ is a randomized oracle algorithm* Dec *such that the following holds for any $x \in \Sigma^k$:*

$$(\forall z \in \Sigma^n \text{ with } d_{\mathrm{H}}(\mathrm{Enc}(x), z) \leq \epsilon n)(\forall i \in [k])\ \Pr[\mathrm{Dec}^z(i) = x_i] \geq 2/3\,,$$

where the probability is over the randomness of the algorithm Dec.

Note that the choice of $2/3$ is arbitrary. Any constant in the range $(\frac{1}{2}, 1)$ would do because we can boost our confidence by running Dec multiple times and outputting the plurality vote. The required number of runs is governed by the following useful proposition.

Proposition 4 (Confidence boosting). *Suppose that a randomized process outputs b with probability $\frac{1}{2} + \eta$. Then the plurality vote of t independent runs equals b with probability at least $1 - \delta$ provided $t \geq \frac{\ln(1/\delta)}{2\eta^2}$.*

Proof. The only way the plurality vote can be different from b is if a subset $S \subset [t]$ of at least $t/2$ of the runs outputs something different than b. The probability that this happens for a fixed S of size s is

$$(\frac{1}{2} - \eta)^s \cdot (\frac{1}{2} + \eta)^{t-s} = \left(\frac{\frac{1}{2} - \eta}{\frac{1}{2} + \eta}\right)^s \cdot (\frac{1}{2} + \eta)^t,$$

which is a non-increasing function of s and is therefore bounded by its value at $s = t/2$, i.e., $(\frac{1}{4} - \eta^2)^{t/2}$. Taking the union bound over all $S \subset [t]$ of size at least $t/2$ shows that the plurality vote equals b except with probability at most

$$2^t(\frac{1}{4} - \eta^2)^{t/2} = (1 - 4\eta^2)^{t/2} \leq \left(\exp(-4\eta^2)\right)^{t/2} = \exp(-2\eta^2 t)\,,$$

where we used the fact that $1 + x \leq \exp(x)$. The resulting bound is at most δ for t as stated. □

For the Hadamard code we can also locally recover any position of the correct *encoding* y by only looking at a small number of positions of a received word z, provided that y and z differ in less than a fraction $1/4$ of the positions. This is called *local correction*.

Definition 5 (Local Corrector). *A local corrector up to relative error ϵ for an (n,k)-code* Enc *over Σ is a randomized oracle algorithm* Cor *such that the following holds for any $y \in \mathrm{Enc}(\Sigma^k)$:*

$$(\forall z \in \Sigma^n \text{ s.t. } d_{\mathrm{H}}(y,z) \le \epsilon n)(\forall j \in [n])\ \Pr[\mathrm{Cor}^z(j) = y_j] \ge 2/3,$$

where the probability is taken over the internal randomness of the algorithm Cor*.*

For systematic codes, a local corrector yields a local decoder with the same parameters. Note that the Hadamard code is systematic because $\mathrm{Enc}(x)_{e_i} = \langle x, e_i\rangle = x_i$. Thus, the following theorem implies a local decoder up to error $\frac{1}{4} - \eta$ for the Hadamard code, where the decoder only makes a constant number of oracle queries for any constant $\eta > 0$.

Theorem 5. *For any $\eta > 0$ there exists a local corrector up to relative error $\frac{1}{4} - \eta$ for the Hadamard code, and the corrector runs in time polynomial in k/η and makes $O(1/\eta)$ oracle queries.*

Proof. Let $y, z \in \{0,1\}^{2^k}$ where $y = \mathrm{Enc}(x)$ is a codeword and $d_{\mathrm{H}}(y,z) \le (\frac{1}{4} - \eta)2^k$. We will view both strings y and z as functions that map an index $a \in \{0,1\}^k$ to the ath position of the string. Thus, $y(a) = \langle x, a\rangle$. By the linearity of the inner product, y is a linear function, so for any $a, b \in \{0,1\}^k$ we have

$$y(b) = \langle x, b\rangle = \langle x, a + (a+b)\rangle = \langle x, a\rangle + \langle x, a+b\rangle = y(a) + y(a+b). \tag{1}$$

For any fixed b, if we pick a uniformly at random, we have that

$$\Pr\left[y(a) \ne z(a)\right] \le \frac{1}{4} - \eta\,, \text{ and} \tag{2}$$

$$\Pr\left[y(a+b) \ne z(a+b)\right] \le \frac{1}{4} - \eta\,, \tag{3}$$

where the probability is taken over the choice of a. This is because (2) is equivalent to the assumed upper bound on $d_{\mathrm{H}}(y,z)$, and (3) follows because $a + b$ is uniformly distributed when a is.

By a union bound, we have that

$$\Pr\left[y(a) \ne z(a) \text{ or } y(a+b) \ne z(a+b)\right] \le \frac{1}{2} - 2\eta\,,$$

and therefore by (1) that

$$\Pr[y(b) = z(a) + z(a+b)] \ge \frac{1}{2} + 2\eta\,.$$

Thus, the following 2-query randomized oracle algorithm, on input $b \in \{0,1\}^k$ an with oracle access to z, correctly returns $y(b)$ with probability at least $\frac{1}{2} + 2\eta$: Pick $a \in_u \{0,1\}^k$ and output $z(a) + z(a+b)$. By running this procedure $O(1/\eta)$ times and taking the majority vote as in Proposition 4, we obtain a local corrector with the stated properties. □

3.3 Reed–Solomon

Reed–Solomon codes are used in practice in DVDs, QR codes, and satellite transmission, for example, and they are very useful in theory, too. They realize the optimal rate vs. distance trade-off, but they require working over a large alphabet $\mathbb{F}_q$.

Given a message $x \in \mathbb{F}_q^k$, we view x as the successive coefficients of a univariate polynomial P_x of degree at most $k-1$, namely $P_x : a \mapsto \sum_{i=1}^{k} x_i a^{i-1}$, and we encode x as the function table of P_x:

$$\mathrm{Enc}(x) = \big(P_x(a)\big)_{a \in \mathbb{F}_q}.$$

Alternately, one can view x as specifying values at k fixed distinct points, and define the encoding as the function table of the unique interpolating polynomial of degree at most $k-1$. The alternate view yields a systematic code with the same codewords but somewhat different encoding and decoding procedures. We will adopt the first view.

The Reed–Solomon code is linear and has a $q \times k$ Vandermonde matrix as its generator. Since a nonzero polynomial of degree at most $k-1$ can have at most $k-1$ zeroes, the distance of the Reed–Solomon code is at least $q-(k-1) = q-k+1$. Thus, the Reed–Solomon code over $\mathbb{F}_q$ is a $[q, k, q-k+1]$-code. The relationship between the rate and the relative distance is optimal, as the following upper bound is satisfied with equality.

Proposition 6 (Singleton bound). *For any (n, k, d)-code, $k + d \leq n + 1$.*

Proof. Consider puncturing the given (n, k, d)-code C in $d-1$ positions. This results in a $(n', k, 1)$-code C' where $n' = n-(d-1) = n-d+1$. Since the number of codewords of C' equals q^k and cannot be larger than $q^{n'}$, we have that $k \leq n' = n-d+1$, which yields the stated inequality. □

There are several efficient decoding algorithms for Reed–Solomon codes that can correct the maximum number of errors. We will discuss one of these algorithms later in the course.

The dual of a Reed–Solomon code turns out to be another Reed–Solomon code. We leave this as an exercise.

Exercise 1. *Prove that the dual code of the $[q, k, q-k+1]_q$ Reed–Solomon code is the $[q, q-k, k+1]_q$ Reed–Solomon code.*

Hint: Let G and H^T be the generator matrices of the Reed–Solomon codes with parameter $[q, k]_q$ and $[q, q-k]_q$, respectively. Show that the columns of G and H^T are pairwise orthogonal by proving the following identity for all $0 \leq i \leq q-2$: $\sum_{a \in \mathbb{F}_q} a^i = 0$, where the arithmetic is over $\mathbb{F}_q$.

3.4 Concatenation

The Reed–Solomon code optimally trades rate and distance, but requires a large alphabet. One way to reduce the size of the alphabet while maintaining good properties is to view the elements of the larger alphabet as messages over a smaller alphabet, and apply a good error correcting code over the smaller alphabet to each of those messages. This is known as code concatenation, where the code over the larger alphabet is referred to as the outer code, and the other one as the inner code.

Definition 6 (Code concatenation). *The concatenation of the (n_{out}, k_{out})-code Enc_{out} over Σ_{out} with the (n_{in}, k_{in})-code Enc_{in} over Σ_{in} where $\Sigma_{out} = \Sigma_{in}^{k_{in}}$ is the code*

$$x \in \Sigma_{out}^{k_{out}} \mapsto \Big(\mathrm{Enc}_{in}(y_i)\Big)_{i=1}^{n_{out}},$$

where y_i denotes the ith component of $\mathrm{Enc}_{out}(x)$.

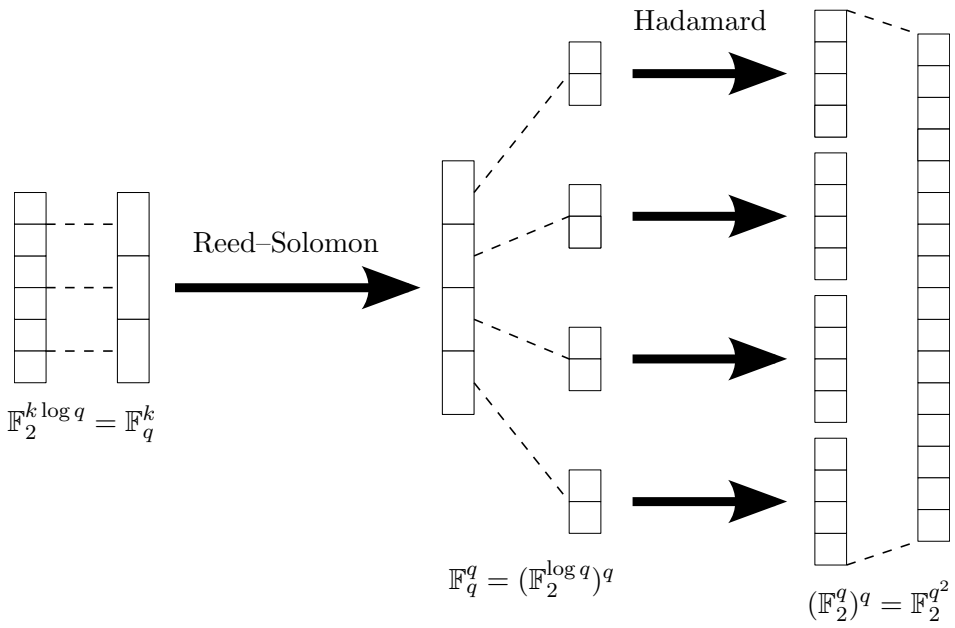

Figure 1: Code concatenation, where the $[q, k]$ Reed–Solomon code serves as the outer code and the $[q, \log q]$ Hadamard code as the inner code, is depicted here with the values $q = 4$ and $k = 3$. The message of the concatenated code is an element of $\mathbb{F}_2^{k \log q}$ and depicted on the very left. The dashed lines indicate that we reinterpret the message as an element of $\mathbb{F}_q^k$, which gets encoded using the Reed–Solomon code into a codeword of length q, where each symbol is an element of $\mathbb{F}_q$. Each symbol is now reinterpreted as an element of $\mathbb{F}_2^{\log q}$, and, in turn, encoded into a codeword of length q over $\mathbb{F}_2$ using the Hadamard code. Finally, the sequence of q codewords of the Hadamard code is concatenated into an element of $\mathbb{F}_2^{q^2}$, the codeword of the concatenated code.

Proposition 7. *The concatenation of an outer $[n_{out}, k_{out}, d_{out}]_{q^{k_{in}}}$-code with an inner $[n_{in}, k_{in}, d_{in}]_q$-code yields an $[n_{out} n_{in}, k_{out} k_{in}, d_{out} d_{in}]_q$-code.*

In particular, concatenating an $[n, k.d]_q$-code where q is a power of two with the Hadamard code yields a binary $[nq, k \log q, dq/2]$-code. Using the $[q, k]$ Reed–Solomon code as the outer code, results in a binary $[q^2, k \log q]$-code with relative distance close to the asymptotically optimal $1/2$. See Figure 1 for an illustration where the outer code is the $[4, 3]_4$ Reed–Solomon code and the inner code the $[4, 2]_2$ Hadamard code, resulting in a $[16, 6]_2$ concatenated code.

Proposition 7 implies that the concatenated code can correct about $d_{out} d_{in}/2$ errors. The following natural decoding procedure corrects half as many: First decode each of the n_{out} blocks according to Enc_{in}, obtaining z_i for $i \in [n_{out}]$, and then decode $z \doteq (z_i)_{i=1}^{n_{out}}$ according to Enc_{out}. If there are fewer than $d_{out} d_{in}/4$ errors, then there are fewer than $d_{out}/2$ blocks with at least $d_{in}/2$ errors; since all the other blocks get decoded correctly by the inner decoding procedure, the outer decoding procedure is guaranteed to succeed.

References

[Rot06] Ron M. Roth. *Introduction to coding theory.* Cambridge University Press, 2006.