

NOTES ON CODE-BASED CRYPTOGRAPHY

GILLES ZÉMOR

1. BACKGROUND ON CODES

1.1. Error-correcting codes, Hamming distance. Let \mathbb{F}_q denote the finite field on q elements. An *error-correcting code*, in all generality, is a just a subset of \mathbb{F}_q^n . A *linear code* is a linear subspace of \mathbb{F}_q^n . We will only be dealing with linear codes, so most of the time we will simply refer to them as codes. The elements of a code are called codevectors or codewords.

Definition 1.1. The *Hamming distance* $d(\mathbf{x}, \mathbf{y})$ between any two vectors $\mathbf{x} = [x_1, \dots, x_n], \mathbf{y} = [y_1, \dots, y_n]$ of \mathbb{F}_q^n is the number of coordinates where \mathbf{x} and \mathbf{y} differ. $d(\mathbf{x}, \mathbf{y}) = \#\{i, x_i \neq y_i\}$.

The function $d(\cdot, \cdot)$ is indeed a distance, it satisfies the triangular inequality

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$$

and also invariance by translation: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x} + \mathbf{z}, \mathbf{y} + \mathbf{z})$. The *weight* of a vector \mathbf{x} , denoted by $|\mathbf{x}|$, is its distance to the zero vector, in other words it is the number of its non-zero coordinates. The *support* of \mathbf{x} is the set of positions i of its non-zero coordinates x_i .

Definition 1.2. The *minimum distance* $d_{\min}(C)$ of a linear code C is the smallest distance between two distinct codewords. It is also equal to the smallest weight of a non-zero codeword.

The *parameters* of a code C are denoted by $[n, k, d]$. The number n is the dimension of the ambient space \mathbb{F}_q^n and is called the length, k is the dimension of the code, $k = \dim_{\mathbb{F}_q} C$, and d is the code minimum distance. If it is unclear what is the finite field \mathbb{F}_q over which C is defined, we may write $[n, k, d]_q$.

Decoding problem: it takes as input a vector $\mathbf{y} \in \mathbb{F}_q^n$ and asks for a codeword \mathbf{c} that minimises the distance $d(\mathbf{c}, \mathbf{y})$ to \mathbf{y} . There may be several solutions to the decoding problem.

Date: September 2024, December 2025.

Error correction. Suppose a codeword \mathbf{c} is corrupted so that some of its coordinates are changed. It is converted to some vector $\mathbf{y} = \mathbf{c} + \mathbf{e}$. The vector \mathbf{e} is called the *error vector*. If the weight of the error vector is not too large, solving the decoding problem with input \mathbf{y} recovers the original codeword \mathbf{c} .

Theorem 1.3. *If $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where \mathbf{c} is a codeword of the code C , and if the error vector \mathbf{e} has weight $< d_{\min}(C)/2$, then the codeword \mathbf{c} is the unique solution to the decoding problem with input \mathbf{y} .*

Proof. Let \mathbf{c}' be a solution to the decoding problem. By the triangular inequality we have

$$d(\mathbf{c}, \mathbf{c}') \leq d(\mathbf{c}, \mathbf{y}) + d(\mathbf{c}', \mathbf{y})$$

but $d(\mathbf{c}', \mathbf{y}) \leq d(\mathbf{c}, \mathbf{y}) < d_{\min}(C)/2$, so we have $d(\mathbf{c}, \mathbf{c}') < d_{\min}(C)$. Therefore we must have $\mathbf{c}' = \mathbf{c}$. \square

Definition 1.4. The matrix \mathbf{G} is said to be a *generator* (or generating) matrix of the code C , if its rows form a basis of C as a vector space. If a generator matrix is of the form $[\mathbf{I}_k \mid \mathbf{A}]$, where \mathbf{I}_k is the $k \times k$ identity matrix, it is said to be in *systematic form*.

1.2. Duality and the syndrome function.

Dual code. The inner product of two vectors $\mathbf{x} = [x_1, \dots, x_n]$ and $\mathbf{y} = [y_1, \dots, y_n]$ is denoted by

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1y_1 + \dots + x_ny_n \in \mathbb{F}_q.$$

The *dual* (or orthogonal) code C^\perp of a code C is defined as

$$C^\perp = \{\mathbf{x} \in \mathbb{F}_q^n, \forall \mathbf{c} \in C, \langle \mathbf{x}, \mathbf{c} \rangle = 0\}.$$

Note that if $\mathbf{G}_1, \dots, \mathbf{G}_k$ are the rows of a generator matrix \mathbf{G} for C , then

$$C^\perp = \{\mathbf{x} \in \mathbb{F}_q^n, \langle \mathbf{x}, \mathbf{G}_i \rangle = 0, i = 1, \dots, k\}.$$

We have:

Proposition 1.5. *For any code C in \mathbb{F}_q^n ,*

$$(1) \quad \dim C + \dim C^\perp = n.$$

To find a generator matrix of the dual code C^\perp given a generator matrix of a code C , the following proposition is useful.

Proposition 1.6. *If $[\mathbf{I}_k \mid \mathbf{A}]$ is a generator matrix for a code C , then $[-\mathbf{A}^\top \mid \mathbf{I}_{n-k}]$ is a generator matrix for C^\perp .*

Parity-check matrix, syndrome function. A generator matrix \mathbf{H} of the dual code C^\perp of a code C is called a *parity-check matrix* of the code C . Given a parity-check matrix \mathbf{H} of the code C , the associated *syndrome* function is defined as:

$$\begin{aligned} \sigma : \mathbb{F}_q^n &\rightarrow \mathbb{F}_q^{n-k} \\ \mathbf{x} &\mapsto \sigma(\mathbf{x}) = \mathbf{H} \mathbf{x}^\top \\ \sigma(\mathbf{x}) &= \begin{bmatrix} \langle \mathbf{H}_1, \mathbf{x} \rangle \\ \cdots \\ \langle \mathbf{H}_{n-k}, \mathbf{x} \rangle \end{bmatrix} \end{aligned}$$

where the \mathbf{H}_i s are the rows of \mathbf{H} . However, one usually prefers to think of the syndrome function as given by the expression:

$$(2) \quad \sigma(\mathbf{x}) = x_1 \mathbf{h}_1 + x_2 \mathbf{h}_2 + \cdots + x_n \mathbf{h}_n$$

where $\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n$ denote the columns of the matrix \mathbf{H} . From the definition of C^\perp and (1), we have $(C^\perp)^\perp = C$, and therefore:

Proposition 1.7. *Let C be a code and σ an associated syndrome function. For $\mathbf{x} \in \mathbb{F}_q^n$, we have $\mathbf{x} \in C$ iff $\sigma(\mathbf{x}) = 0$.*

Dual form of the decoding problem: it takes as input an element $\mathbf{s} \in \mathbb{F}_q^{n-k}$ of the syndrome space, and asks for a vector $\mathbf{e} \in \mathbb{F}_q^n$ of minimum weight such that $\sigma(\mathbf{e}) = \mathbf{s}$. This form of the decoding problem is also referred to as the *syndrome decoding problem*.

Note that the two forms of the decoding problem are really equivalent. If we can solve the syndrome version and we want to find the closest codeword to \mathbf{y} , we compute $\mathbf{s} = \sigma(\mathbf{y})$ and then solve the syndrome decoding problem to find \mathbf{e} of minimum weight such that $\sigma(\mathbf{e}) = \mathbf{s}$. Then we set $\mathbf{c} = \mathbf{y} - \mathbf{e}$: since $\sigma(\mathbf{c}) = \sigma(\mathbf{y}) - \sigma(\mathbf{e}) = 0$ Proposition 1.7 implies that \mathbf{c} is a codeword and it must be a solution to the decoding problem. Conversely, if we are given $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and we want to find \mathbf{e} of minimum weight such that $\sigma(\mathbf{e}) = \mathbf{s}$, then we can first find some arbitrary solution \mathbf{y} to the system of linear equations $\sigma(\mathbf{y}) = \mathbf{s}$ (without any weight requirements), which is algorithmically simple linear algebra, and then we solve the decoding problem in its original form to find a codeword \mathbf{c} closest to \mathbf{y} , after which $\mathbf{e} = \mathbf{y} - \mathbf{c}$ is the required solution to the syndrome decoding problem with input \mathbf{s} .

Example 1.8. The Hamming code of length 7. Let

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

and define C to be the binary code (code over \mathbb{F}_2) defined by the parity-check matrix \mathbf{H} . Proposition 1.7 together with the expression (2) of the syndrome imply that words of weight 1, 2 cannot be codewords, since all columns of \mathbf{H} are non-zero and distinct from each other. There are several (how many?) triples of columns that sum to zero, so the code minimum distance is 3. The parameters of the code are $[7, 4, 3]$. Furthermore, since every non-zero element of \mathbb{F}_2^3 is a column of \mathbf{H} , we have that every possible input $\mathbf{y} \notin C$ to the decoding problem for C has a solution that is at distance 1 to \mathbf{y} .

2. REED-SOLOMON CODES

2.1. Narrow-sense Reed-Solomon codes.

Definition 2.1. Let $1 \leq k \leq n \leq q$, and let $\alpha_1, \dots, \alpha_n$ be distinct elements of \mathbb{F}_q . We denote $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]$. The narrow-sense Reed-Solomon code over \mathbb{F}_q defined by $\boldsymbol{\alpha}$ and k is the set of n -tuples

$$[f(\alpha_1), f(\alpha_2), \dots, f(\alpha_n)]$$

where $f(X)$ ranges over all polynomials of $\mathbb{F}_q[X]$ of degree $< k$.

A Reed-Solomon code C is clearly a linear code. Since a non-zero polynomial of degree $< k \leq n$ cannot have n roots, the map

$$f(X) \mapsto [f(\alpha_1), f(\alpha_2), \dots, f(\alpha_n)]$$

is injective, and the dimension of C is that of the space of polynomials of degree $< k$, namely $\dim C = k$. By the same number of roots argument, a non-zero codeword of C has at most $k - 1$ zero coordinates, so its weight is at least $n - k + 1$. Therefore $d_{\min}(C) \geq n - k + 1$. The following theorem tells us that this inequality must actually be an equality.

Theorem 2.2. (*Singleton bound*). *The parameters $[n, k, d]$ of a code satisfy the inequality*

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

Proof. Consider the linear map

$$\begin{aligned} C &\rightarrow \mathbb{F}_q^{k-1} \\ [x_1, \dots, x_n] &\mapsto [x_1, \dots, x_{k-1}]. \end{aligned}$$

It takes a space of dimension k (the code C) to a space of dimension $k - 1$, so it has a non-zero kernel. A non-zero vector of the kernel is a codeword with $(k - 1)$ zero coordinates, so it has weight $\leq n - k + 1$. \square

A code satisfying the above Singleton bound is called an *MDS* code. They are optimal, meaning that any code with the same length and dimension cannot have a larger minimum distance. Altogether we have just proved:

Theorem 2.3. *The Reed-Solomon code over \mathbb{F}_q defined by α and k , denoted hereafter by $RS_k(\alpha)$, has parameters*

$$[n, k, d = n - k + 1]$$

and is therefore an MDS code.

Let us mention two useful facts on MDS codes.

Proposition 2.4. *Let \mathbf{G} be a $k \times n$ generator matrix of an $[n, k, d]$ code C . The code C is an MDS code iff every $k \times k$ submatrix of \mathbf{G} is non-singular.*

Proof. A vector \mathbf{c} of C has weight $< n - k + 1$ iff it has at least k coordinates equal to zero. This happens iff some non-trivial linear combination of the rows of \mathbf{G} is zero on k coordinates. But this means exactly that some $k \times k$ submatrix of \mathbf{G} has some linear combination of its rows equal to zero, i.e. is singular over \mathbb{F}_q . \square

Theorem 2.5. *The dual of an MDS code is an MDS code.*

Proof. Let C be an MDS code of length n and dimension k with generator matrix \mathbf{G} . The dual code C^\perp has therefore \mathbf{G} as a parity-check matrix. Let σ be the associated syndrome function defined by the matrix \mathbf{G} . Since every $k \times k$ submatrix of \mathbf{G} is non-singular, every non-trivial linear combination of at most k columns of \mathbf{G} is non-zero. From Proposition 1.7 we therefore have that any codeword of C^\perp must have weight at least $k + 1$, which means that C^\perp is MDS since it has dimension $n - k$. \square

Before generalising the definition of Reed-Solomon codes, let us mention:

Proposition 2.6. *The matrix*

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix}$$

is a generator matrix for the Reed-Solomon code $RS_k(\alpha)$.

Proof. The rows of the matrix correspond to the evaluation of the polynomials $1, X, X^2, \dots, X^{k-1}$. \square

2.2. Generalised Reed-Solomon codes. We generalise slightly the definition of a Reed-Solomon code. Let $\alpha = [\alpha_1, \dots, \alpha_n]$ and $\beta = [\beta_1, \dots, \beta_n]$, where the α_i are *distinct* elements of \mathbb{F}_q as before, and where the β_i are in \mathbb{F}_q , but *not necessarily distinct*, and are *non-zero*.

Definition 2.7. The (generalised) Reed-Solomon code over \mathbb{F}_q defined by k , $1 \leq k \leq n \leq q$, and by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, denoted by $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$, is the set of n -tuples

$$[\beta_1 f(\alpha_1), \beta_2 f(\alpha_2), \dots, \beta_n f(\alpha_n)]$$

where $f(X)$ ranges over all polynomials of $\mathbb{F}_q[X]$ of degree $< k$. From now on we refer to generalised Reed-Solomon codes simply as Reed-Solomon codes. Note that the narrow-sense Reed-Solomon code $RS_k(\boldsymbol{\alpha})$ is equal to $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$, with $\boldsymbol{\beta} = (1, 1, \dots, 1)$. All previous arguments carry over straightforwardly and we now have:

Theorem 2.8. *The RS codes $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ have parameters $[n, k, d = n - k + 1]$.*

Proposition 2.9. *The matrix*

$$\mathbf{G} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_n \\ \beta_1 \alpha_1 & \beta_2 \alpha_2 & \cdots & \beta_n \alpha_n \\ \beta_1 \alpha_1^2 & \beta_2 \alpha_2^2 & \cdots & \beta_n \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ \beta_1 \alpha_1^{k-1} & \beta_2 \alpha_2^{k-1} & \cdots & \beta_n \alpha_n^{k-1} \end{bmatrix}$$

is a generator matrix for the Reed-Solomon code $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

The following notion will be useful to find the dual of an RS code.

2.3. Star products. If $\mathbf{x} = [x_1, \dots, x_n], \mathbf{y} = [y_1, \dots, y_n] \in \mathbb{F}_q^n$, let us define the coordinate-wise product of \mathbf{x} and \mathbf{y} :

$$\mathbf{x} * \mathbf{y} = [x_1 y_1, x_2 y_2, \dots, x_n y_n].$$

To lighten notation, instead of $\mathbf{x} * \mathbf{y}$ we will simply write \mathbf{xy} , which, in our context, is unlikely to be confused with some other product structure. Notice that we have:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{xy}, \mathbf{1} \rangle$$

where $\mathbf{1}$ denotes the all-one vector $\mathbf{1} = [1, 1, \dots, 1]$. More generally we have

$$(3) \quad \langle \mathbf{xz}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{yz} \rangle.$$

Remark 2.10. Notice that the generator matrix \mathbf{G} of the RS code $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ given by Proposition 2.9 has rows that may be conveniently thought of as the geometric progression $\boldsymbol{\beta}, \boldsymbol{\beta}\boldsymbol{\alpha}, \boldsymbol{\beta}\boldsymbol{\alpha}^2, \dots, \boldsymbol{\beta}\boldsymbol{\alpha}^{k-1}$.

Definition 2.11. If B and C are two codes of length n over \mathbb{F}_q , we define their star product (also called Hadamard product, or Schur product) $B * C$ as the *linear span* of all products \mathbf{bc} , for all $\mathbf{b} \in B, \mathbf{c} \in C$.

Again we will mostly write BC instead of $B * C$ and, similarly, use the shorthand C^2 to denote $C * C$. Context should dispel any confusion with Cartesian products.

Remark 2.12. If B and C are two codes of length n and dimensions k and ℓ respectively, and if $\mathbf{b}_1, \dots, \mathbf{b}_k$ and $\mathbf{c}_1, \dots, \mathbf{c}_\ell$ are bases of B and C respectively, then the code BC is generated by the set of products $\mathbf{b}_i \mathbf{c}_j$ of basis vectors, $i = 1, \dots, k$ $j = 1, \dots, \ell$. In particular we have

$$\dim B * C \leq \dim B \dim C$$

though the actual dimension of $B * C$ may sometimes be much smaller.

Finally, notice that we have:

Proposition 2.13. *If $B = RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $C = RS_\ell(\boldsymbol{\alpha}, \boldsymbol{\gamma})$, then*

$$B * C = BC = RS_{k+\ell-1}(\boldsymbol{\alpha}, \boldsymbol{\beta}\boldsymbol{\gamma})$$

(with the convention that if $k + \ell - 1 > n$, then $RS_{k+\ell-1}(\)$ is defined as the whole space).

2.4. The dual of a Reed-Solomon code.

Theorem 2.14. *The dual of a Reed-Solomon code $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is a Reed-Solomon code $RS_{n-k}(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ for some vector $\boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_n]$.*

Proof. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be fixed. Consider the Reed-Solomon code $C_{n-1} = RS_{n-1}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of dimension $n - 1$. Let us set $\boldsymbol{\gamma}$ to be a non-zero codeword of the dual code C_{n-1}^\perp . Since C_{n-1} has dimension $n - 1$, the vector $\boldsymbol{\gamma}$ is uniquely defined up to multiplication by a non-zero element of \mathbb{F}_q . Since C_{n-1} is MDS its dual is MDS (Theorem 2.5), which means that $\boldsymbol{\gamma}$ has weight n , i.e. all the γ_i are non-zero. It makes sense therefore to talk about Reed-Solomon codes $RS_k(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ and our choice of $\boldsymbol{\gamma}$ means that we have $RS_{n-1}(\boldsymbol{\alpha}, \boldsymbol{\beta})^\perp = RS_1(\boldsymbol{\alpha}, \boldsymbol{\gamma})$. Equivalently, from Remark 2.10 we have

$$\langle \boldsymbol{\beta}\boldsymbol{\alpha}^i, \boldsymbol{\gamma} \rangle = 0 \quad \text{for all } i = 0 \dots n - 2.$$

Applying (3) to the above for $i = 1, \dots, n - 2$, we obtain

$$\langle \boldsymbol{\beta}\boldsymbol{\alpha}^i, \boldsymbol{\gamma}\boldsymbol{\alpha} \rangle = 0 \quad \text{for all } i = 0 \dots n - 3.$$

We therefore have that both $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}\boldsymbol{\alpha}$ are orthogonal to $RS_{n-2}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, meaning that $RS_{n-2}(\boldsymbol{\alpha}, \boldsymbol{\beta})^\perp = RS_2(\boldsymbol{\alpha}, \boldsymbol{\gamma})$. Continuing in this way we obtain that $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})^\perp = RS_{n-k}(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ for every k . \square

Computing $\boldsymbol{\gamma}$ from $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is straightforward linear algebra (Gaussian elimination). It may be tedious by hand though, but is simple in some cases, for example:

– EXERCICE 1. *Let $n = q - 1$ and $\alpha_1, \dots, \alpha_n$ be the non-zero elements of \mathbb{F}_q .*

- (1) *Show that $\alpha_1 + \dots + \alpha_n = 0$.*
- (2) *Show that $\alpha_1^i + \dots + \alpha_n^i = -1$ if $i = 0 \pmod n$ and $\alpha_1^i + \dots + \alpha_n^i = 0$ otherwise.*

- (3) Show that if $\beta = \mathbf{1}$, then $\gamma = \alpha$. More generally, show that we have $\gamma = \alpha\beta^{-1}$ (all coordinates of β are non-zero, so it is invertible for the star product).

2.5. Decoding Reed-Solomon codes. Let C be the $RS_k(\alpha, \beta)$ Reed-Solomon code over \mathbb{F}_q , for some α, β . Let \mathbf{c} be a codeword and suppose that it is corrupted in t positions, so that we are given a vector $\mathbf{y} \in \mathbb{F}_q^n$, such that $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $|\mathbf{e}| = t$. We suppose furthermore that $t < d_{\min}/2$, equivalently $t \leq (n - k)/2$. The goal is to recover \mathbf{c} with an algorithm of reasonable complexity.

The key to the decoding algorithm is to introduce an auxiliary Reed-Solomon code, which is $L = RS_{t+1}(\alpha, \mathbf{1}) = RS_{t+1}(\alpha)$ and is called the *error-locator* code (or simply locator code). Note from Proposition 2.13 that the star product $\Pi = CL$ is the Reed-Solomon code $RS_{k+t}(\alpha, \beta)$.

Note that there exists a non-zero codeword ℓ of L that is zero on the t positions in error, i.e. the support of \mathbf{e} . This is simply because L has been chosen of dimension $t + 1$. Alternatively, we can think of such a codeword as being defined by a polynomial $f(X)$ of degree t that is zero on the values α_i that define the support of \mathbf{e} . For such a vector ℓ we therefore have, from $\mathbf{y} = \mathbf{c} + \mathbf{e}$ and $\mathbf{e}\ell = 0$,

$$\mathbf{y}\ell = \mathbf{c}\ell + \mathbf{e}\ell = \mathbf{c}\ell.$$

Therefore, by definition of the star product, we have $\mathbf{y}\ell \in \Pi$. So the first step of the decoding procedure is to *localise the errors*. That means find $\ell \in L$ of weight $n - t$ such that $\mathbf{y}\ell = 0$. We now know that such a ℓ is a solution of $\mathbf{y}\ell \in \Pi$. This is simply a linear system. Concretely, we compute $\mathbf{y}\ell_0, \mathbf{y}\ell_1, \dots, \mathbf{y}\ell_t$ where $\ell_0, \ell_1, \dots, \ell_t$ is a fixed basis of the code L . We write

$$(4) \quad \ell = \lambda_0\ell_0 + \lambda_1\ell_1 + \dots + \lambda_t\ell_t$$

and define the syndrome function σ for the code Π (by means of a parity-check matrix of Π that can be precomputed). The condition $\mathbf{y}\ell \in \Pi$ gets therefore rewritten as:

$$\sigma(\mathbf{y}\ell) = \lambda_0\sigma(\mathbf{y}\ell_0) + \lambda_1\sigma(\mathbf{y}\ell_1) + \dots + \lambda_t\sigma(\mathbf{y}\ell_t) = 0.$$

So we solve this linear system in the variables $\lambda_0, \lambda_1, \dots, \lambda_t$ which gives the required value of ℓ defined by (4).

Now we know that among the solutions to the above linear system there must be one ℓ that is zero on the positions in error. But could there be other “parasite” solutions? The answer is no, because if $\mathbf{y}\ell \in \Pi$, then from $\mathbf{y} = \mathbf{c} + \mathbf{e}$ and $\mathbf{c}\ell \in \Pi$ we have $\mathbf{e}\ell \in \Pi$. But Π is a Reed-Solomon code of dimension $k + t$, and therefore of minimum distance $n - k - t + 1$ which is greater than t by our assumption $t \leq (n - k)/2$. So because $|\mathbf{e}\ell| \leq |\mathbf{e}| = t$, it must be the case that $\mathbf{e}\ell = 0$. Therefore, any non-zero solution ℓ to $\mathbf{y}\ell \in \Pi$ will be a vector with exactly t zero

coordinates (there can't be more because the minimum distance of the RS code L is $n - t$), which correspond exactly to the positions in error.

Summarising:

Decoding algorithm for $C = RS_k(\alpha, \beta)$.

Input: $\mathbf{y} = \mathbf{c} + \mathbf{e}$, $|\mathbf{e}| = t$.

Set $\ell_0 = \mathbf{1}, \ell_1 = \alpha, \dots, \ell_t = \alpha^t$.

Define a syndrome function σ for the code $\Pi = RS_{k+t}(\alpha, \beta)$.

Compute $\sigma(\mathbf{y}\ell_0), \sigma(\mathbf{y}\ell_1), \dots, \sigma(\mathbf{y}\ell_t)$.

Solve the linear system $\lambda_0\sigma(\mathbf{y}\ell_0) + \lambda_1\sigma(\mathbf{y}\ell_1) + \dots + \lambda_t\sigma(\mathbf{y}\ell_t) = 0$ in the indeterminates $\lambda_i \in \mathbb{F}_q$ and choose any non-zero solution.

Compute the codeword $\boldsymbol{\ell} = [\ell_1, \ell_2, \dots, \ell_n] = \lambda_0\ell_0 + \lambda_1\ell_1 + \dots + \lambda_t\ell_t$ of L .

For every i such that $\ell_i \neq 0$, declare the position i to be error-free, i.e. set $c_i = y_i$.

Let $E = \{i, \ell_i = 0\}$. This is the set of positions in error. To recover the missing coordinates $c_i, i \in E$, solve the linear system in the indeterminates c_i

$$\sum_{i \in E} c_i \mathbf{h}_i + \sum_{i \notin E} y_i \mathbf{h}_i = 0$$

where $\mathbf{h}_1, \dots, \mathbf{h}_n$ are the columns of a parity-check matrix \mathbf{H} of C .

Output $\mathbf{c} = [c_1, \dots, c_n]$.

Remark 2.15. For the above algorithm we have implicitly supposed that the number t of errors is known to the decoder, which will be the typical cryptography setting. However it may often be that t is only known to be an upper bound on the actual number of errors. In this case the algorithm runs without any changes, and the same arguments prove that it yields the correct result. What will happen is that there will be more available solutions $\lambda_0, \dots, \lambda_t$ to the linear system, but any non-zero solution will still work: it will also be possible that $|E| < t$, and that some values of $i \in E$ will turn out to also be error-free coordinates for \mathbf{c} , i.e. $c_i = y_i$.

3. GOPPA CODES

3.1. Alternant codes. Suppose that we want codes over \mathbb{F}_q of length n significantly larger than q . Then the Reed-Solomon construction does not work because it is restricted to $n \leq q$ (actually the Reed-Solomon construction generalises to $n = q + 1$, but this makes little difference). This is typically the case for small

values of q , notably for $q = 2$. We don't necessarily have to forget about the Reed-Solomon construction altogether though: what we can do is take a Reed-Solomon code C of length n over an extension field \mathbb{F}_{q^m} of \mathbb{F}_q , and then consider the subcode of C made up of those codewords whose coordinates all fall into \mathbb{F}_q . This is sometimes called a *subfield subcode* construction. Applied to Reed-Solomon codes, this gives us the class of so-called *alternant codes*.

Definition 3.1. An *alternant code* over \mathbb{F}_q is the set of codewords of a Reed-Solomon code over \mathbb{F}_{q^m} , for some $m > 1$, whose coordinates all belong to \mathbb{F}_q .

An alternant code is usually specified by a parity-check matrix of the underlying Reed-Solomon code, so an $r \times n$ matrix \mathbf{H} over \mathbb{F}_{q^m} of the form

$$\begin{bmatrix} \gamma \\ \gamma\alpha \\ \dots \\ \gamma\alpha^{r-1} \end{bmatrix}$$

with $\alpha = [\alpha_1, \dots, \alpha_n]$ and $\gamma = [\gamma_1, \dots, \gamma_n]$, $\alpha_i, \gamma_i \in \mathbb{F}_{q^m} \setminus \{0\}$. The alternant code is then

$$C = \{\mathbf{c} \in \mathbb{F}_q^n, \mathbf{H}\mathbf{c}^\top = 0\}.$$

Decomposing the entries of \mathbf{H} over an \mathbb{F}_q -basis of \mathbb{F}_{q^m} , we get that the r rows of \mathbf{H} become rm rows over \mathbb{F}_q , in other words C becomes defined by rm linear equations over \mathbb{F}_q . We therefore have:

Proposition 3.2. *The alternant code defined by the $r \times n$ matrix \mathbf{H} over \mathbb{F}_{q^m} is an \mathbb{F}_q -linear code of dimension $k \geq n - rm$.*

In all generality we only have an inequality for the dimension k because we have no guarantee that the rm linear equations over \mathbb{F}_q will be linearly independent.

Since the alternant code defined by \mathbf{H} is contained in a Reed-Solomon code, its minimum distance is at least that of the RS code. In other words:

Proposition 3.3. *The alternant code defined by the $r \times n$ matrix \mathbf{H} over \mathbb{F}_{q^m} is an \mathbb{F}_q -linear code of minimum distance $d \geq r + 1$.*

Finally, alternant codes come with a natural decoding algorithm, they can be decoded simply by solving the decoding problem for the ambient Reed-Solomon code. If we are decoding $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $|\mathbf{e}| < (r + 1)/2$, then we know that \mathbf{c} is the unique solution to the decoding problem with input \mathbf{y} and that the Reed-Solomon decoder will find it.

3.2. Goppa codes, a first definition. Goppa codes are a particular subclass of the family of alternant codes, though that is not immediately apparent from the construction below.

Let q be fixed and set $Q = q^m$. Let $\alpha_1, \dots, \alpha_n$ be some fixed, distinct elements of \mathbb{F}_Q and let $G(X) \in \mathbb{F}_Q[X]$ be some polynomial of degree r such that $G(\alpha_i) \neq 0$ for $i = 1, \dots, n$. Now for any vector $\mathbf{a} = (a_1, \dots, a_n)$ over \mathbb{F}_q , we define the rational function in the variable X ,

$$(5) \quad R_{\mathbf{a}}(X) = \sum_{i=1}^n \frac{a_i}{X - \alpha_i}.$$

Note that the $(X - \alpha_i)$ are invertible modulo $G(X)$ since we have chosen $G(X)$ such that $G(\alpha_i) \neq 0$. We may therefore consider the quantity $R_{\mathbf{a}}(X)$ in the quotient ring $\mathbb{F}_Q[X]/G(X)$. We define:

Definition 3.4. For $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]$ and $G(X)$, the Goppa code $\Gamma(\boldsymbol{\alpha}, G)$ consists of all vectors $\mathbf{a} \in \mathbb{F}_q^n$ such that

$$R_{\mathbf{a}}(X) = 0$$

in the ring $\mathbb{F}_Q[X]/G(X)$.

It should be obvious that Γ defined above is a linear code over \mathbb{F}_q . We now compute a convenient parity-check matrix (over \mathbb{F}_Q) for Γ .

3.3. An \mathbb{F}_Q -parity-check matrix for Goppa codes.

Theorem 3.5. Let $\boldsymbol{\alpha}$ and $G(X)$ define the Goppa code $\Gamma = \Gamma(\boldsymbol{\alpha}, G)$. Let

$$\mathbf{H} = \begin{bmatrix} \gamma \\ \gamma \boldsymbol{\alpha} \\ \dots \\ \gamma \boldsymbol{\alpha}^{r-1} \end{bmatrix}$$

with $\boldsymbol{\gamma} = [G(\alpha_1)^{-1}, G(\alpha_2)^{-1}, \dots, G(\alpha_n)^{-1}]$. Then $\Gamma = \{\mathbf{a} \in \mathbb{F}_q^n, \mathbf{H}\mathbf{a}^\top = 0\}$. In particular Γ is an alternant code.

Proof. First we express the inverse of $(X - \alpha_i)$ modulo $G(X)$ as a polynomial of degree $< r$. For this we need only write:

$$(X - \alpha_i)^{-1} = -\frac{G(X) - G(\alpha_i)}{X - \alpha_i} G(\alpha_i)^{-1},$$

since it is easily checked that multiplying the right-hand side by $(X - \alpha_i)$ gives 1. So by definition of Γ , $\mathbf{a} \in \Gamma$ iff

$$(6) \quad \sum_{i=1}^n a_i \frac{G(X) - G(\alpha_i)}{X - \alpha_i} G(\alpha_i)^{-1} = 0$$

in $\mathbb{F}_Q[X]$, since the left-handside of (6) is a polynomial of degree $< r$. Writing $G(X) = g_r X^r + g_{r-1} X^{r-1} + \dots + g_1 X + g_0$, where $g_i \in \mathbb{F}_Q$ and $g_r \neq 0$, we apply

the formula

$$\frac{X^j - \alpha_i^j}{X - \alpha_i} = X^{j-1} + \alpha_i X^{j-2} + \dots + \alpha_i^{j-1}$$

with the purpose of making the left-hand side of (6) more explicit, to obtain:

$$\frac{G(X) - G(\alpha_i)}{X - \alpha_i} = g_r(X^{r-1} + \alpha_i X^{r-2} + \dots + \alpha_i^{r-1}) + \dots + g_2(X + \alpha_i) + g_1$$

Equating the coefficients of $X^{r-1}, \dots, X, 1$ to zero in (6), we get that $\mathbf{a} \in \Gamma$ iff $\mathbf{M}\mathbf{a}^\top = 0$ with

$$\mathbf{M} = \begin{bmatrix} g_r G(\alpha_1)^{-1} & \dots & g_r G(\alpha_n)^{-1} \\ (g_{r-1} + \alpha_1 g_r) G(\alpha_1)^{-1} & \dots & (g_{r-1} + \alpha_n g_r) G(\alpha_n)^{-1} \\ \dots & \dots & \dots \\ (g_1 + \alpha_1 g_2 + \dots + \alpha_1^{r-1} g_r) G(\alpha_1)^{-1} & \dots & (g_1 + \alpha_n g_2 + \dots + \alpha_n^{r-1} g_r) G(\alpha_n)^{-1} \end{bmatrix}$$

Now \mathbf{M} gets rewritten as:

$$\mathbf{M} = \begin{bmatrix} g_r & 0 & 0 & \dots & 0 \\ g_{r-1} & g_r & 0 & \dots & 0 \\ g_{r-2} & g_{r-1} & g_r & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ g_1 & g_2 & g_3 & \dots & g_r \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \dots & \dots & \dots & \dots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{bmatrix} \mathbf{D}$$

with \mathbf{D} being the diagonal matrix

$$\mathbf{D} = \begin{bmatrix} G(\alpha_1)^{-1} & & & 0 \\ & G(\alpha_2)^{-1} & & \\ & & \ddots & \\ 0 & & & G(\alpha_n)^{-1} \end{bmatrix}.$$

Since the leftmost matrix in the decomposition of \mathbf{M} is square and invertible, we have that $\mathbf{M}\mathbf{a}^\top = 0$ iff $\mathbf{H}\mathbf{a}^\top = 0$ where

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \dots & \dots & \dots & \dots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{bmatrix} \mathbf{D}$$

which is exactly the parity-check matrix claimed by the Theorem. \square

So why the convoluted definition of Section 3.2 if they can be defined by the characterisation of Theorem 3.5? Because the original definition allows us to sometimes obtain information that would be difficult to obtain from the parity-check matrix of Theorem 3.5. We now illustrate.

3.4. Binary Goppa codes. Let $q = 2$ and $Q = 2^m$. Define a Goppa code $\Gamma = \Gamma(\boldsymbol{\alpha}, G)$ and suppose $G(X)$ is chosen to be squarefree, meaning that for every irreducible polynomial $P(X)$ of $\mathbb{F}_Q[X]$, we have $P(X)^2 \nmid G(X)$. Let $\mathbf{a} = [a_1, \dots, a_n]$ be a (binary) codeword in Γ and let $I_{\mathbf{a}} = \{i, a_i = 1\}$ be the support of \mathbf{a} . Define the polynomial:

$$f_{\mathbf{a}}(X) = \prod_{i \in I_{\mathbf{a}}} (X - \alpha_i)$$

We have:

$$f'_{\mathbf{a}}(X) = \sum_{i \in I_{\mathbf{a}}} \prod_{\substack{j \in I_{\mathbf{a}} \\ j \neq i}} (X - \alpha_j)$$

and (5) gets rewritten as:

$$R_{\mathbf{a}}(X) = \sum_{i \in I_{\mathbf{a}}} \frac{1}{X - \alpha_i} = \frac{f'_{\mathbf{a}}(X)}{f_{\mathbf{a}}(X)}$$

Since $G(\alpha_i) \neq 0$ for every i , we have that $f_{\mathbf{a}}(X)$ is invertible modulo $G(X)$, so

$$R_{\mathbf{a}}(X) = 0 \pmod{G(X)} \quad \text{iff} \quad G(X) \mid f'_{\mathbf{a}}(X).$$

Now the field \mathbb{F}_Q is of characteristic 2, so $f'_{\mathbf{a}}(X)$ is a sum of monomials of even degree, so it is a sum of squares (recall that every element of \mathbb{F}_Q is a square), hence it is a square in $\mathbb{F}_Q[X]$. Therefore $G(X) \mid f'_{\mathbf{a}}(X)$ iff $G(X)^2 \mid f'_{\mathbf{a}}(X)$. Summarising:

Proposition 3.6. *If the Goppa polynomial $G(X)$ is squarefree, then, over the binary field we have:*

$$\Gamma = \Gamma(\boldsymbol{\alpha}, G(X)) = \Gamma(\boldsymbol{\alpha}, G(X)^2).$$

So, setting $r = \deg G(X)$, if we think of Γ as defined by $G(X)$, then Proposition 3.6 tells us that the minimum distance of Γ is at least $2r + 1$ as opposed to $r + 1$ guaranteed by Proposition 3.3. Alternatively, if we think of Γ as defined by $G(X)^2$, then Proposition 3.6 tells us that the dimension of Γ is at least $n - rm$ as opposed to $n - 2rm$ guaranteed by Proposition 3.2.

To decode Γ we should use its structure given by $\Gamma(\boldsymbol{\alpha}, G(X)^2)$, since Reed-Solomon decoding is then guaranteed to decode correctly up to r errors.

4. THE ORIGINAL McELIECE CRYPTOSYSTEM

4.1. The McEliece paradigm. McEliece proposed a very general method for devising a *public-key* encryption scheme. Let $\mathcal{M} = \mathbb{F}_2^k$ be the space of plain messages that we want to encrypt, in other words we assume our plaintexts to be k -bit strings. Choose an error-correcting code C of length n and dimension k and

publish an arbitrary generator matrix \mathbf{G} for this code. The matrix \mathbf{G} is the public key and is used for encryption. To encrypt a message $\mathbf{m} \in \mathbb{F}_2^k$, we compute

$$\mathbf{y} = \mathbf{mG} + \mathbf{e}$$

where \mathbf{e} is a *random* binary vector of length n and some fixed weight t . Notice that \mathbf{mG} is a codeword for the code C , so the ciphertext \mathbf{y} is a noisy version of a codeword.

The secret key gives access to a hidden decoding algorithm that is able to efficiently remove error vectors of weight t . So to decipher the ciphertext \mathbf{y} , apply the decoding algorithm to remove the error \mathbf{e} and obtain \mathbf{mG} . Writing $\mathbf{G} = [\mathbf{A} \mid \mathbf{B}]$ and assuming, without loss of generality, \mathbf{A} to be invertible, we have $\mathbf{mG} = [\mathbf{mA} \mid \mathbf{mB}]$ and we may multiply on the right \mathbf{mA} by \mathbf{A}^{-1} to recover \mathbf{m} .

The working assumption is that without knowledge of the hidden data that allows one to decode noisy codewords of C , the adversary that tries to decrypt is faced with a generic instance of the decoding problem, which is generally assumed to be intractable (more on that later).

The assumption for a McEliece cryptosystem is therefore generally formulated as *indistinguishability* from a random code. To make this slightly more formal, what we require is a family \mathcal{F} of codes C , of length n and dimension k say, that come with a low-complexity decoding algorithm from errors of weight t . We assume:

Indistinguishability assumption. Consider the following two ways of constructing a matrix G :

- (1) Choose a random code C from the family \mathcal{F} , and choose a random $k \times n$ generator matrix G for C ,
- (2) Choose a uniformly random $k \times n$ matrix G .

There should not exist an algorithm with complexity less than some security parameter (e.g. that uses less than 2^{80} arithmetic operations), that given as input a matrix G that has been randomly constructed either according to method 1 or method 2, outputs (1) or (2) with a success probability significantly better than $1/2$ (a random guess).

We now remark that the indistinguishability assumption implies that deciphering random encrypted messages is not feasible without the secret or some extra knowledge. Indeed, if there were an algorithm that decrypts, then, by definition of the McEliece scheme, such an algorithm decodes from t errors random codewords of a member C of \mathcal{F} ; the indistinguishability assumption implies therefore that this algorithm also decodes from t errors random codewords of a *random code* (because if the algorithm behaves differently on a random code, then it is a distinguisher!)

If we are convinced that decoding random codes of length n and dimension k from t errors is infeasible, then the indistinguishability assumption implies security. We

could envisage having security without indistinguishability, but cryptographers like to err on the side of caution.

4.2. The original McEliece cryptosystem. How can we implement the above idea practically? McEliece’s proposal was to use binary Goppa codes $\Gamma(\boldsymbol{\alpha}, G(X))$. So the parameters of the Goppa code are chosen, namely m , the degree of the extension field \mathbb{F}_Q of \mathbb{F}_2 , the length n , and the degree r of the Goppa polynomial. Then a random vector $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]$, with distinct $\alpha_i \in \mathbb{F}_Q$, is chosen. Finally one chooses a random squarefree polynomial $G(X) \in \mathbb{F}_Q[X]$ of degree r that does not evaluate to 0 on any of the α_i . the public-key is an arbitrary (random) generator matrix G for the code C . Everything else about the code is kept secret.

We recall that the matrix \mathbf{H} in Theorem 3.5 defines a Reed-Solomon code and that its dual is a Reed-Solomon code $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ with $k = n - r$. We can think of $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the *parent* Reed-Solomon code of the Goppa code $\Gamma(\boldsymbol{\alpha}, G(X)$, i.e. $\Gamma(\boldsymbol{\alpha}, G(X))$ is the set of binary vectors that belong to $RS_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$. The secret key consists of therefore of $\boldsymbol{\alpha}, \boldsymbol{\beta}$ which enables us to decode both the parent Reed-Solomon code and the Goppa code $\Gamma(\boldsymbol{\alpha}, G(X))$ by applying the decoding algorithm of Section 2.5. Summarising:

The original McEliece cryptosystem Choose parameters m, n, r . Choose $\boldsymbol{\alpha} = [\alpha_1, \alpha_n]$, where the α_i are distinct elements of \mathbb{F}_Q , $Q = 2^m$. Choose a random squarefree polynomial $G(X) \in \mathbb{F}_Q[X]$ of degree r . Define the binary Goppa code $C = \Gamma(\boldsymbol{\alpha}, G(X))$.

Public key: a random generator matrix G of C

Secret key: $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ that define the Reed-Solomon code $RS_{n-r}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ parent to C .

Encryption: for a message $\mathbf{m} \in \mathbb{F}_2^k$, construct the ciphertext as

$$\mathbf{y} = \mathbf{mG} + \mathbf{e}$$

where $\mathbf{e} \in \mathbb{F}_2^n$ is a random binary vector of weight r .

Decryption: Use $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to apply the decoding algorithm of Section 2.5 and recover $\mathbf{c} = \mathbf{mG}$ from \mathbf{y} . Recover \mathbf{m} from \mathbf{mG} by linear algebra (solve a linear system).

Remark 4.1. One doesn’t really need the knowledge of $\boldsymbol{\beta}$ to decode, the vector $\boldsymbol{\alpha}$ is sufficient. Indeed, in the decoding algorithm of Section 2.5 one can define the code Π as the star-product of L with the code over \mathbb{F}_Q generated by the matrix G , which will be a subcode of (and probably simply equal to) the product of L with the parent Reed-Solomon code $RS_{n-r}(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

McEliece originally proposed to use the parameters $m = 10$, $n = 1024$ and $r = 50$. So the Goppa code C has parameters $[1024, k \geq 524, d \geq 101]$ and the decoding algorithm allows the decoding of any pattern of 50 errors. In practice the dimension k of the code actually equals 524.

5. RANDOM BINARY LINEAR CODES

5.1. The minimum distance of random codes: the Gilbert-Varshamov bound. There are several relatively natural ways of choosing a code randomly. If one wishes for a code of length n and dimension k , then we can

- choose C uniformly among all $[n, k]$ codes,
- choose a uniform random $k \times n$ matrix \mathbf{G} and declare it to be a generator matrix for C ,
- choose a uniform random $(n - k) \times n$ matrix \mathbf{H} and declare it to be a parity-check matrix for C .

These probability distributions are almost the same but not quite: with the first method we always get a code of dimension k , but with the second method we may get a code of dimension less than k , and with the third method we may get a code of dimension more than k . It is worth remarking that in practice, if we wanted a uniformly distributed $[n, k]$ code we would choose a random uniform $k \times n$ generator matrix, compute its rank and throw it away and start again if its rank were not k . Since every $[n, k]$ code has the same number of generator matrices, this is guaranteed to yield the uniform distribution. Furthermore, it is not very difficult to show (see Lemma 5.1 below) that a random $k \times n$ binary matrix has rank k with probability at least $1 - 1/2^{n-k}$. For large n and $1 \ll k \ll n$, in practice we would therefore never see the difference between the three methods of choosing the random code C . (The analysis after Lemma 5.1 makes this more specific).

Choosing a random uniform $[n, k]$ code C is possibly the most natural way, but it is the least convenient for computing probabilities: below we choose a random code C by choosing a random uniform $r \times n$ matrix, giving us typically an $[n, k]$ code with dimension $k = n - r$.

Switching from one distribution to another. The following lemma is quite useful when studying properties of random matrices and the associated codes.

Lemma 5.1. *Let \mathbf{H} be an $r \times n$ uniformly random binary matrix, $n > r$. It holds that*

$$\mathbb{P}[\text{rk}(\mathbf{H}) < r] \leq \frac{1}{2^{n-r}}$$

where $\text{rk}(\mathbf{H})$ denotes the rank of \mathbf{H} over \mathbb{F}_2 .

Proof. Let us write $x = n - r$. We use induction on r . For $r = 1$ we have that $\text{rk}(\mathbf{H}) < 1$ if and only if \mathbf{H} is the zero row which occurs with probability $1/2^{1+x} < 1/2^x$.

Let \mathbf{H} be an $r \times (r+x)$ matrix and suppose the statement holds for $(r-1) \times y$ matrices for any y . Let \mathbf{H}_{r-1} be the matrix made up of the first $r-1$ rows of \mathbf{H} . We can write:

$$\begin{aligned} \mathbb{P}[\text{rk}(\mathbf{H}) < r] &= \mathbb{P}[\text{rk}(\mathbf{H}_{r-1}) < r-1] \\ &\quad + \mathbb{P}[\text{rk}(\mathbf{H}) < r \mid \text{rk}(\mathbf{H}_{r-1}) = r-1] \mathbb{P}[\text{rk}(\mathbf{H}_{r-1}) = r-1] \\ &\leq \mathbb{P}[\text{rk}(\mathbf{H}_{r-1}) < r-1] + \mathbb{P}[\text{rk}(\mathbf{H}) < r \mid \text{rk}(\mathbf{H}_{r-1}) = r-1]. \end{aligned}$$

By the induction hypothesis we have $\mathbb{P}[\text{rk}(\mathbf{H}_{r-1}) < r-1] \leq 1/2^{1+x}$. Furthermore, the probability that the last row belongs to a given subspace of dimension at most $r-1$ is not more than

$$\mathbb{P}[\text{rk}(\mathbf{H}) < r \mid \text{rk}(\mathbf{H}_{r-1}) = r-1] \leq \frac{2^{r-1}}{2^{r+x}} = \frac{1}{2^{1+x}}$$

hence, $\mathbb{P}[\text{rk}(\mathbf{H}) < r] \leq 1/2^{1+x} + 1/2^{1+x} \leq 1/2^x$. \square

Corollary 5.2. *Let P_U be the uniform distribution on the set \mathcal{H} of $r \times n$ matrices \mathbf{H} , for $r < n$. Let P_r be the distribution on \mathcal{H} induced by the uniform distribution on the subset of matrices of \mathcal{H} that have rank r . Let A be a subset of matrices of \mathcal{H} . We have*

$$\mathbb{P}_U[A] \leq \mathbb{P}_r[A] \left(1 + \frac{1}{2^{n-r} - 1} \right).$$

Proof. Let R be the event equal to the set of matrices \mathbf{H} of \mathcal{H} of rank r . Lemma 5.1 states that $\mathbb{P}_U[R] \geq 1 - 1/2^{n-r}$. Furthermore, since $\mathbb{P}_r[A] = \mathbb{P}_U[A \mid R]$, we can apply Bayes' Theorem to obtain that

$$\mathbb{P}_r[A] = \mathbb{P}_U[A \mid R] = \frac{\mathbb{P}[R \mid A] \mathbb{P}[A]}{\mathbb{P}[R]} \leq \frac{\mathbb{P}[A]}{1 - \frac{1}{2^{n-r}}} \leq \mathbb{P}[A] \left(1 + \frac{1}{2^{n-r} - 1} \right).$$

\square

Let $B_{n,t}$ be the Hamming ball of radius t centered on the zero vector, centre excluded. In other words $B_{n,t}$ is the union of all the non-zero vectors of weight t or less. We have:

$$(7) \quad |B_{n,t}| = \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}.$$

Theorem 5.3 (Gilbert-Varshamov lower bound). *Let n, r, t , be such that $r, t < n$ and $|B_{n,t}| < 2^r$. With probability at least $1 - |B_{n,t}|/2^r$ the random code C defined by a uniform random $r \times n$ parity-check matrix \mathbf{H} has minimum distance $d > t$.*

Proof. Let \mathbf{x} be a non-zero vector. The syndrome of \mathbf{x} for the random matrix \mathbf{H} is clearly uniformly distributed in \mathbb{F}_2^r and is therefore equal to zero with probability $1/2^r$. Let $X_{\mathbf{x}}$ be the Bernoulli random variable equal to 1 if $\mathbf{x} \in C$ and 0 otherwise.

We have $E[X_{\mathbf{x}}] = P[\mathbf{x} \in C] = 1/2^r$. Let

$$(8) \quad X = \sum_{\mathbf{x} \in B_{n,t}} X_{\mathbf{x}}.$$

The random variable X equals the number of non-zero vectors of weight t or less that fall into the random code C . Therefore, $d_{\min}(C) > t$ if and only if $X = 0$. By linearity of expectation we have

$$E[X] = \sum_{\mathbf{x} \in B_{n,t}} E[X_{\mathbf{x}}] = \frac{|B_{n,t}|}{2^r}$$

whence the result, since $P[X \neq 0] \leq E[X]$ and

$$P[d_{\min}(C) > t] = P[X = 0] = 1 - P[X \neq 0].$$

□

For example, for $n = 1024$ and $r = 500$, namely the original McEliece parameters, computing $|B_{n,t}|$ with (7) for $t = 100$ (with sage) and applying Theorem 5.3 we get that a random $[1024, 524]$ code has minimum distance ≥ 101 with probability more than $1 - 3/10^{10}$. Applying the theorem with $t = 110$ gives us that there exists $[1024, 524]$ codes with minimum distance at least 111.

There is a sort of converse to Theorem 5.3, namely:

Theorem 5.4 (Gilbert-Varshamov upper bound). *Let n, r, t , be such that $r, t < n$ and $|B_{n,t}| > 2^r$. With probability at least $1 - 2^r/|B_{n,t}|$ the random code C defined by a uniform random $r \times n$ parity-check matrix \mathbf{H} has minimum distance $d \leq t$.*

Proof. We define as before X by (8), i.e. the random variable that counts the number of non-zero codewords of the random code C that fall into the Hamming ball $B_{n,t}$ of radius t . We have as before $E[X] = |B_{n,t}|/2^r$. Now Chebyshev's inequality gives us that:

$$(9) \quad P[X = 0] \leq \frac{\text{Var}[X]}{E[X]^2}.$$

The Bernoulli variables $X_{\mathbf{x}}$ are easily seen to be pairwise independent, therefore the variance of X satisfies

$$\text{Var}[X] = |B_{n,t}| \text{Var}[X_{\mathbf{x}}] < |B_{n,t}| E[X_{\mathbf{x}}] = E[X]$$

since $\text{Var}[X_{\mathbf{x}}] = E[X_{\mathbf{x}}] - E[X_{\mathbf{s}}]^2$. Inequality (9) gives us therefore

$$P[X = 0] < \frac{1}{E[X]}$$

which gets rewritten as $P[X \neq 0] > 1 - 2^r/|B_{n,t}|$ which is the result, since $X > 0$ means exactly that $d_{\min}(C) \leq t$. □

Together, Theorems 5.3 and 5.4 give us that the minimum distance of a random code defined by a random $r \times n$ parity-check matrix \mathbf{H} is, for large n , very probably very close to the value t for which $\binom{n}{t}$ is closest to 2^r .

Choosing n even and $r = n/2$, one can show that this value of t equals $cn + o(n)$ with c a constant equal to 0.11002.... So a random code of large length n and dimension $n/2$ has very probably a minimum distance very close to 11% of the code length.

5.2. Information-set decoding. If a code is indistinguishable from a random code, a question naturally arises: how easy or how difficult is it to decode a random code? In other words, how do we decode a linear code generically, i.e. without knowledge of any particular additional structure for the code. The most efficient generic approach for decoding that we know of is to use *information set decoding*.

Let C be an $[n, k]$ code. Let \mathbf{H} be a parity-check matrix for C , and assume that \mathbf{H} is in *systematic* form, which means that

$$\mathbf{H} = [\mathbf{B} \ \mathbf{I}_r]$$

with \mathbf{I}_r the $r \times r$ identity matrix, $r = n - k$. Let us assume for the simplicity of the analysis to follow that $k = r = n/2$, which corresponds to the original McEliece setting.

Now let \mathbf{e} be a vector of weight t : suppose for a moment that the support of \mathbf{e} is entirely included in the right half of the set of coordinates $[n]$, i.e. $\text{supp}(\mathbf{e}) \subset [n/2 + 1, \dots, n]$. Then the syndrome $\sigma(\mathbf{e})$ of \mathbf{e} is a sum of columns of \mathbf{I}_r , i.e. a sum of columns of weight 1, therefore $|\mathbf{e}| = t$. Conversely, the decoding problem, i.e. the task of writing $\sigma(\mathbf{e})$ as a sum of t columns of \mathbf{H} is then trivial, one simply writes it as a sum of columns of weight 1. How likely is this easy decoding event E_t when \mathbf{e} is a uniform random vector of weight t ? We can compute its probability exactly, it is:

$$\begin{aligned} \mathbb{P}[E_t] &= \frac{\binom{n/2}{t}}{\binom{n}{t}} \\ &= \frac{n/2(n/2 - 1) \cdots (n/2 - t + 1)}{n(n - 1) \cdots (n - t + 1)} \\ &= \frac{1}{2^t} \frac{n(n - 2)(n - 4) \cdots (n - 2t + 2)}{n(n - 1)(n - 2) \cdots (n - t + 1)} \\ &= \frac{1}{2^t} \left(1 - \frac{1}{n - 1}\right) \left(1 - \frac{2}{n - 2}\right) \cdots \left(1 - \frac{t - 1}{n - t + 1}\right) \end{aligned}$$

so that

$$\mathbb{P}[E_t] = \frac{1}{2^t} + o(1)$$

when $t = o(\sqrt{n})$. So the event E_t is not very likely but not impossibly unlikely if t is not too large. But what can we do when the event E_t does not occur? Well, we can use another matrix \mathbf{H}' of C in systematic form. Specifically, we choose a random subset $J \subset [n]$, $|J| = n/2$: this defines the square $n/2 \times n/2$ submatrix \mathbf{A}_J whose columns equal the columns of \mathbf{H} indexed by J and in the same order. We then compute $\mathbf{H}' = \mathbf{H}_J = \mathbf{A}_J^{-1}\mathbf{H}$ which has the same rowspace as \mathbf{H} (namely C^\perp) and equals the identity matrix when restricted to the columns indexed by J . It may happen that the matrix \mathbf{A}_J is not invertible, but let's ignore that objection for now. Assuming that the set J is chosen uniformly of size $n/2$, the probability that a fixed error vector \mathbf{e} of weight t is such that $\text{supp}(\mathbf{e}) \subset J$ is:

$$\begin{aligned} \text{P}[\text{supp}(\mathbf{e}) \subset J] &= \frac{\binom{n-t}{n/2-t}}{\binom{n}{n/2}} = \frac{\frac{(n-t)!}{(n/2-t)!(n/2)!}}{\frac{n!}{(n/2)!^2}} = \frac{(n-t)!(n/2)!}{(n/2-t)!n!} \\ &= \frac{\frac{(n/2)!}{(n/2-t)!t!}}{\frac{n!}{(n-t)!t!}} = \frac{\binom{n/2}{t}}{\binom{n}{t}} \end{aligned}$$

which is the same probability as that of the event E_t that we computed before when J was fixed and \mathbf{e} was random, and is approximately $1/2^t$ when t is small enough.

Note that we can compute the syndrome $\sigma_J(\mathbf{e})$ for the matrix \mathbf{H}_J from the syndrome $\sigma(\mathbf{e})$ of \mathbf{e} for \mathbf{H} . This is simply $\sigma_J(\mathbf{e}) = \mathbf{A}_J^{-1}\sigma(\mathbf{e})$.

So now we have a randomised algorithm for decoding error vectors \mathbf{e} of weight t : it goes as follows.

Prange's algorithm for decoding an error vector of weight t .

Input: an $r \times n$ parity-check matrix \mathbf{H} for the code C and the syndrome $\sigma(\mathbf{e}) = \mathbf{H}\mathbf{e}^\top$ of a vector \mathbf{e} of weight t .

Choose a random subset $J \subset [n]$ of size r such that the square submatrix \mathbf{A}_J of \mathbf{H} , obtained by restricting the columns of \mathbf{H} to the indices in J , is invertible.

find the corresponding parity-check matrix $\mathbf{H}_J = \mathbf{A}_J^{-1}\mathbf{H}$ of C which is the identity matrix on the column index set J ,

compute the corresponding syndrome $\sigma_J(\mathbf{e}) = \mathbf{A}_J^{-1}\sigma(\mathbf{e})$ of \mathbf{e} for the matrix \mathbf{H}_J ,

check whether the syndrome weight $|\sigma_J(\mathbf{e})|$ equals t .

If yes, then output the vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_n]$ such that

- $\varepsilon_i = 1$ if $i \in J$ and the i -th coordinate of $\sigma_J(\mathbf{e})$ equals 1,
- $\varepsilon_i = 0$ otherwise.

If no, repeat with another random J .

Note that if the original error vector \mathbf{e} has weight less than half the code minimum distance, we are guaranteed that the solution $\hat{\mathbf{e}}$ found by the decoder equals \mathbf{e} .

Running time. How many choices of J do we need for the decoder to output a solution? We need the decoder to find a J that contains the support $\text{supp}(\mathbf{e})$ of the error vector. Ignoring the issue of the invertibility of the A_J , and assuming that the decoder chooses the subsets J uniformly and independently of each other, we have that the decoder outputs the solution after exactly m attempts with probability approximately

$$\left(1 - \frac{1}{2^t}\right)^{m-1} \frac{1}{2^t}.$$

In other words, the number m of attempts needed to output the solution follows a geometric distribution with parameter $1/2^t$, and its expectation is therefore 2^t . Consequently, the expected running time of the algorithm is 2^t times the complexity of Gaussian elimination needed to compute \mathbf{H}_J .

What about the objection that \mathbf{A}_J may not be invertible? This can indeed happen frequently. But what Gaussian elimination does is transform \mathbf{H} progressively so as to make columns of weight 1 appear one after the other. Gaussian elimination will only get stuck when the number of columns of weight 1 it has constructed equals the rank of the submatrix \mathbf{A}_J . Now if \mathbf{H} is uniform random, so is \mathbf{A}_J , and it can be shown that the rank of a uniform random $r \times r$ binary matrix is less than $r - h$ with probability $\leq 1/2^h$ *. So if we just halt Gaussian elimination when it can't go further we will have a matrix \mathbf{H}_J which is almost the identity matrix on the column coordinate set J , and the probability that the corresponding syndrome is of weight t will drop below $1/2^t$, but not in a significant way. We therefore do not deviate very much from our above average complexity analysis. In practice one will simply modify slightly the set J when Gaussian elimination stalls, so as to obtain a full $r \times r$ identity submatrix of \mathbf{H}_J .

Comment. The decoding strategy we have just described is called *information set* decoding, because when a parity-check matrix is of the form $\mathbf{H} = [\mathbf{B} \ \mathbf{I}_r]$, the first $k = n - r$ indices are called an information set. This is because the restriction (or puncturing) map $C \rightarrow \mathbb{F}_2^k$ is bijective. The first k coordinates of a codeword $[c_1, \dots, c_n]$ can be thought of as the *information* symbols, and the remaining symbols are redundant symbols whose purpose is the protection of the information symbols. The inverse map $\mathbb{F}_2^k \rightarrow C$ is the encoding map, it takes the k -tuple $\mathbf{m} = [m_1, \dots, m_k]$ to the n -tuple $\mathbf{c} = [\mathbf{m}, \mathbf{m}\mathbf{B}^\top] \in C$.

*For example because this probability is less than the probability that a random $r \times (r + h)$ matrix has rank less than r , and Lemma 5.1 applies.

There are a number of algorithmic improvements that can be brought to information set decoding but we do not discuss them here: known improvements do not change the exponential (in t) nature of the decoding complexity nor the value of the exponent for $t = o(\sqrt{n})$.

Looking for codewords of small weight. Suppose we do not wish to solve the decoding problem but instead wish to find a word of weight t in a code C , t being possibly the code minimum distance. We can easily adapt the information set decoding technique. Once a parity-check matrix is in systematic form, it will not happen that the support $\text{supp}(\mathbf{c})$ of the desired codeword is included in C (there is no such codeword!), but it may happen that exactly one element i of the support falls outside J . When this happens, the corresponding column \mathbf{h}_i of \mathbf{H}_J has weight $t - 1$, since \mathbf{h}_i must be the sum of $t - 1$ identity matrix columns. For a random set $J \subset [n]$, a similar computation as before gives that the desirable event happens with probability approximately $t/2^t$. Finding a codeword of weight t can therefore be achieved with a similar complexity as that of decoding from t errors.

6. MCELIECE WITH MDPC CODES

The McEliece cryptosystem with Goppa codes has withstood the test of time: it was first proposed in 1978 and a serious weakness has yet to be found. However, its security is based on a very ad hoc assumption, which is the indistinguishability of a random Goppa code defined by a random generator matrix with a uniform random code. Goppa codes have a lot of hidden algebraic structure, and it is difficult to be truly confident that this structure cannot be made apparent with an efficient algorithm.

A very interesting alternative to using Goppa codes in the McEliece cryptosystem is to use so called *MDPC* codes. The abbreviation MDPC stands for *Moderate Density Parity Check*, a variation of the well-known class of *Low-Density Parity Check* (LDPC) codes.

Loosely speaking, an LDPC code is a code that can be defined by a sparse parity-check matrix \mathbf{H} . This means that the code length is assumed to be large ($n \geq 1000$, say) and rows and columns of \mathbf{H} have their weights bound from above by a small constant: a row or column weight that behaves like $\log n$ is sometimes tolerated.

An MDPC code is a binary code that is defined by an $r \times n$ parity-check matrix \mathbf{H} which is sparse, but not too sparse. The requirement is that the weight of every column is equal to a constant w . The column weight is a constant in the sense that it is the same for every column, but it is required to grow with n : typically one chooses w to scale as \sqrt{r} . It is also required that the row weight is constant which must then be equal to wn/r . For simplicity we shall assume that $r = n/2$, which is what is used in practice to create an McEliece-like cryptosystem.

As we shall see below, an MDPC code comes with a simple decoding algorithm that can correct approximately $r/4w$ random errors. As we have seen, correcting this number of errors without knowledge of the MDPC structure has a complexity equal to $2^{r/(4w)}$ matrix inversions when using information set decoding: on the other hand, recovering the hidden structure amounts to finding the parity-check equations of weight $2w$, i.e. finding words of weight $2w$ in the dual code, which has a complexity of 2^{2w} matrix inversions, also with information set techniques. It makes sense therefore to choose w so that those two complexities are roughly equal, which means

$$(10) \quad 2w = r/(4w),$$

which translates into $2w = \frac{1}{2}\sqrt{n}$. This is the typical value of w chosen for MDPC-McEliece. The hidden structure of the MDPC code is assumed to be much looser than that of a Goppa code, uncovering it essentially consists of finding (somewhat) low-weight codewords in the dual code.

Decoding MDPC codes: the bit-flip algorithm. Let \mathbf{H} be the sparse $r \times n$ parity-check matrix of the MDPC code C , $n = 2r$. Let $\mathbf{h}_1, \dots, \mathbf{h}_n$ be the columns of \mathbf{H} and assume for simplicity that they have been chosen independently with the uniform distribution among vectors of weight w in \mathbb{F}_2^r . This will not give a constant row weight for \mathbf{H} , but the decoder analysis will be believable enough.

Let \mathbf{e} be an error vector of weight t , and consider its syndrome $\mathbf{s} = \sigma(\mathbf{e})$. We have

$$\mathbf{s} = \sum_{i \in \text{supp}(\mathbf{e})} \mathbf{h}_i.$$

Let us think of the syndrome computation as a process where we are adding columns \mathbf{h}_i of \mathbf{H} one after the other. Let $J \subsetneq \text{supp}(\mathbf{e})$, and denote the corresponding ‘‘partial’’ syndrome $\mathbf{s}_J = \sum_J \mathbf{h}_i$. If $|J|$ is not too large, what we expect when summing some new column \mathbf{h}_j , $j \notin J$, to \mathbf{s}_J , is that the supports of the low-weight column vectors \mathbf{s}_J and \mathbf{h}_j have a small intersection and therefore that $|\mathbf{s}_J + \mathbf{h}_j| \gg |\mathbf{s}_J|$. Conversely, if $j \in J$ we expect to have $|\mathbf{s}_J + \mathbf{h}_j| \ll |\mathbf{s}_J|$.

This suggests therefore the following decoding algorithm:

Bit-flip decoding (one iteration).

Input: the syndrome $\mathbf{s} = \mathbf{H}\mathbf{e}^\top$ of an error-vector of weight $t \leq 2w$.

For every $j \in (n]$, compute $\mathbf{s} + \mathbf{h}_j$,

- if $|\mathbf{s} + \mathbf{h}_j| \leq |\mathbf{s}|$, set $\varepsilon_j = 1$,
- if $|\mathbf{s} + \mathbf{h}_j| > |\mathbf{s}|$, set $\varepsilon_j = 0$.

If $\sigma(\boldsymbol{\varepsilon}) = \mathbf{s}$, output $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_n]$.

If $\sigma(\boldsymbol{\varepsilon}) \neq \mathbf{s}$, we may either declare a decoding failure, or set $\mathbf{s} := \mathbf{s} + \sigma(\boldsymbol{\varepsilon})$ and repeat with another iteration of the algorithm. We shall just analyse one iteration of the decoder.

Since $t \leq 2w$, the syndrome weight $|\mathbf{s}|$ is at most $2wt$ which, by equation (10) is at most $r/4$. Let $j \notin \text{supp}(\mathbf{e})$: let us estimate the probability that the algorithm, which is trying to compute $\varepsilon_j = e_j = 0$, makes a wrong decision. This will happen if at least half the support of \mathbf{h}_j falls into the support of \mathbf{s} . The probability that exactly half the support of \mathbf{h}_j is in $\text{supp}(\mathbf{s})$ equals:

$$\begin{aligned} \mathbb{P}[|\mathbf{s} + \mathbf{h}_j| = |\mathbf{s}|] &= \frac{\binom{\frac{r}{4}}{\frac{w}{2}} \binom{\frac{3r}{4}}{\frac{w}{2}}}{\binom{r}{w}} \leq \frac{w!}{\left(\frac{w}{2}\right)! \left(\frac{w}{2}\right)!} \frac{\left(\frac{1}{4}r\right)^{w/2} \left(\frac{3}{4}r\right)^{w/2}}{(r-w)^w} \\ &\leq 2^w \left(\frac{\sqrt{3}}{4}\right)^w \left(\frac{r}{r-w}\right)^w \\ &\leq \left(\frac{\sqrt{3}}{2}\right)^w \left(1 + \frac{w}{r-w}\right)^w. \end{aligned}$$

Applying (10) we have

$$\left(1 + \frac{w}{r-w}\right)^w = \left(1 + \frac{1}{\frac{r}{w} - 1}\right)^w = \left(1 + \frac{1}{8w - 1}\right)^w \leq \left(1 + \frac{1}{7w}\right)^w$$

which is bounded from above by a small constant c . We have therefore

$$\mathbb{P}[|\mathbf{s} + \mathbf{h}_j| \leq |\mathbf{s}|] \leq cw \left(\frac{\sqrt{3}}{2}\right)^w.$$

The w term in front of the exponential is actually pessimistic and can be replaced by a constant. In any case, we have that this is a vanishingly small probability for large n and proves that one iteration of the bit-flipping algorithm should correct up to $2w$ errors with vanishingly small probability of failure.

7. BREAKING WITH THE McELIECE PARADIGM: CRYPTOGRAPHY WITH RANDOM CODES

We will now introduce public-key encryption schemes that are provably as hard to break as it is to decode random binary linear codes. First, we need to introduce a decisional version of the decoding problem. Let n, k, t be integers, $k, t < n$, and let us make the following hypothesis, that we will refer to as $\mathcal{H}(n, k, t)$:

Hypothesis $\mathcal{H}(n, k, t)$. *There does not exist an algorithm \mathcal{A} of reasonable complexity that*

- (1) takes as input a uniform random $k \times n$ binary matrix \mathbf{A} , together with a binary vector \mathbf{y} of length n that is either
- (i) $\mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in C$ is a uniformly random codeword of the code C generated by the rows of \mathbf{A} , and \mathbf{e} is a uniformly random vector of weight t in \mathbb{F}_2^n , independent of \mathbf{c}
 - (ii) or is a uniformly random vector \mathbf{u}
- (2) outputs whether \mathbf{y} was produced according to distribution (i) or distribution (ii) with a non-negligible advantage over just randomly guessing (i) or (ii).

As discussed in Section 5, we can replace the choice of a uniform random matrix \mathbf{A} by a random matrix that is uniform among matrices of rank k .

An equivalent form of the hypothesis $\mathcal{H}(n, k, t)$ is its dual form, which we can formulate as:

Dual form of hypothesis $\mathcal{H}(n, k, t)$. *There does not exist an algorithm \mathcal{A} of reasonable complexity that*

- (1) takes as input a uniform random $(n - k) \times n$ binary matrix \mathbf{B} , together with a binary vector \mathbf{s} of length $n - k$ that is either
- (i) equal to $\mathbf{B}\mathbf{e}^\top$, where \mathbf{e} is a uniformly random vector of weight t in \mathbb{F}_2^n ,
 - (ii) or is a uniformly random vector $\mathbf{u} \in \mathbb{F}_2^{n-k}$
- (2) outputs whether \mathbf{y} was produced according to distribution (i) or distribution (ii) with a non-negligible advantage over just randomly guessing (i) or (ii).

We will see later that the hypothesis $\mathcal{H}(n, k, t)$ actually reduces to the difficulty of decoding random codes. More precisely, if an algorithm \mathcal{A} that distinguishes significantly between (i) and (ii) did exist, then we can convert it into an algorithm \mathcal{A}' , that is not much more complex than \mathcal{A} , that takes as input a vector $\mathbf{y} = \mathbf{c} + \mathbf{e}$ produced according to distribution (ii), and that outputs \mathbf{c} and \mathbf{e} with a significant probability of success.

7.1. The Alekhovich cryptosystem. Let \mathbf{A} be a random $k \times n$ matrix and let $\boldsymbol{\varepsilon}$ be a random vector in \mathbb{F}_2^n of weight t . It will be important that we set the parameter t to be significantly smaller than \sqrt{n} . Let \mathbf{H} be the $(k + 1) \times n$ matrix obtained by appending to \mathbf{A} an additional row consisting of the vector

$$\mathbf{y} = \mathbf{x}\mathbf{A} + \boldsymbol{\varepsilon}$$

where \mathbf{x} is a uniform vector in \mathbb{F}_2^k independent of $\boldsymbol{\varepsilon}$. In other words, \mathbf{y} is the sum of the error vector $\boldsymbol{\varepsilon}$ and a random codeword of the code generated by the matrix \mathbf{A} .

Let C be the code with parity-check matrix \mathbf{H} . The public key is a generating matrix for the code C . In this cryptosystem the message space consists of a single bit $\mathcal{M} = \{0, 1\}$.

Encryption. To encrypt 0, output

$$\mathcal{C}(0) = \mathbf{c} + \mathbf{e}$$

where \mathbf{c} is a random codeword of C and \mathbf{e} is a random vector of weight t . To encrypt 1, output

$$\mathcal{C}(1) = \mathbf{u}$$

where \mathbf{u} is a uniform random vector of \mathbb{F}_2^n .

Decryption. The secret key is the vector $\boldsymbol{\varepsilon}$. To decrypt the encrypted bit $\mathcal{C}(m)$, $m \in \{0, 1\}$, compute

$$b = \langle \boldsymbol{\varepsilon}, \mathcal{C}(m) \rangle$$

and declare b to be the decrypted plaintext. We have that

$$\begin{aligned} \langle \boldsymbol{\varepsilon}, \mathcal{C}(0) \rangle &= \langle \boldsymbol{\varepsilon}, \mathbf{c} + \mathbf{e} \rangle = \langle \boldsymbol{\varepsilon}, \mathbf{c} \rangle + \langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle \\ &= \langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle \end{aligned}$$

because $\boldsymbol{\varepsilon}$, being a row of the parity-check matrix \mathbf{H} , is orthogonal to all codewords of C . Since we have imposed on both $\boldsymbol{\varepsilon}$ and \mathbf{e} to be of weight $t \ll \sqrt{n}$, the probability that $\langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle = 0$ is close to 1. On the other hand, since $\mathcal{C}(1)$ is a random vector, we have that when $m = 1$, decryption succeeds with probability exactly 1/2. To obtain a reliable cryptosystem, we can encrypt the secret bit m several times: an multiple encryption of 0 will be decrypted mostly as 0, while an encryption of 1 will be decrypted as 1 roughly half of the time.

Of course, this cryptosystem is hugely impractical. But it comes with a reward in the form of a security proof. Breaking the system is impossible without violating hypothesis \mathcal{H} .

Security reduction. Suppose there exists a decryption algorithm \mathcal{D} that extracts m from $\mathcal{C}(m)$ given only knowledge of the code C (and not $\boldsymbol{\varepsilon}$). We will use the hypothesis \mathcal{H} twice to derive a contradiction in two steps.

First step. The decryption algorithm \mathcal{D} should work without any noticeable difference if the code C is replaced by a random code C' defined by a uniform random parity-check matrix \mathbf{H}' , i.e. if the vector $\mathbf{y} = \mathbf{x} + \boldsymbol{\varepsilon}$ used in defining the last row of \mathbf{H} is uniformly random, equivalently if $\boldsymbol{\varepsilon}$ is taken to be uniformly random rather than random of weight t . If there were a noticeable difference, this would yield a way of distinguishing whether \mathbf{y} is uniformly random or at distance t from the random code generated by the matrix \mathbf{A} , contradicting hypothesis $\mathcal{H}(n, k, t)$.

Second step. We may now suppose that we are using the encryption scheme with the random code C' rather than the original code C . Again, the decryption algorithm \mathcal{D} should work just as well when the error vector \mathbf{e} used to encrypt the message $m = 0$ is replaced by a uniform random vector. Otherwise we again have a distinguisher between a uniform random vector and a vector of the form $\mathbf{c} + \mathbf{e}$ where \mathbf{e} is of weight t and \mathbf{c} is a random codeword of the *random* code C' . Again

this would contradict the decisional decoding hypothesis. (To be precise, since we are defining the random code through a random parity-check matrix rather than a random generating matrix, we are applying hypothesis $\mathcal{H}(n, n - (k + 1), t)$). But we now have an absurd result, which is that the decryption algorithm \mathcal{D} should somehow be able to decrypt in the situation when the encryption of both 0 and 1 are uniform random vectors of \mathbb{F}_2^n , which clearly cannot be achieved with a success probability different from $1/2$.

7.2. Regev-like variant. We now introduce a slight improvement on the Alekhnovich cryptosystem: it still encodes a single bit of plaintext with a long string of bits, but it improves upon the decryption failure probability.

We choose the same parameters as before. We let again \mathbf{A} a random $k \times n$ matrix and let $\boldsymbol{\varepsilon}$ be a random vector of \mathbb{F}_2^n of weight $t \ll n^{1/2}$. Let \mathbf{x} be a uniform random vector in \mathbb{F}_2^k and let

$$\mathbf{y} = \mathbf{x}\mathbf{A} + \boldsymbol{\varepsilon}.$$

The message (plaintext) space is $\mathcal{M} = \{0, 1\}$ and the matrix \mathbf{A} and the vector \mathbf{y} make up the public key. We take the secret (decryption) key to be \mathbf{x} .

Encryption. To encrypt $m \in \mathbb{F}_2$, choose a random t -weight vector \mathbf{e} of \mathbb{F}_2^n and compute $\mathcal{C}_1 = \mathbf{A}\mathbf{e}^\top$. Then compute $\mathcal{C}_2 = m + \langle \mathbf{y}, \mathbf{e} \rangle$. The encryption of m is defined to be the pair:

$$\mathcal{C}(m) = (\mathcal{C}_1, \mathcal{C}_2).$$

Decryption. Compute

$$m' = \mathcal{C}_2 + \langle \mathbf{x}, \mathcal{C}_1 \rangle.$$

We have $\langle \mathbf{y}, \mathbf{e} \rangle = \mathbf{x}\mathbf{A}\mathbf{e}^\top + \langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle$. Since $\langle \mathbf{x}, \mathcal{C}_1 \rangle = \mathbf{x}\mathbf{A}\mathbf{e}^\top$ we therefore have

$$m' = m + \langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle.$$

Since $\boldsymbol{\varepsilon}$ and \mathbf{e} have been chosen to be of weight $t \ll \sqrt{n}$, it is highly probable that $\langle \boldsymbol{\varepsilon}, \mathbf{e} \rangle = 0$ and $m' = m$.

Security reduction. Suppose there exists a decryption algorithm \mathcal{D} that extracts m from $\mathcal{C}(m)$ given only knowledge of the code \mathbf{A} and \mathbf{y} . We again proceed in two steps, applying hypothesis \mathcal{H} at each step.

Step 1. This is the same as before. We replace the vector \mathbf{y} by a vector that is not chosen to be at distance t from the row space of \mathbf{A} but is a uniformly random vector \mathbf{y}' in \mathbb{F}_2^n . The decryption algorithm should not be able to tell the difference between \mathbf{y} and \mathbf{y}' , otherwise we would violate hypothesis $\mathcal{H}(n, k, t)$, so it continues to decrypt m from the input $\mathbf{A}, \mathbf{y}', \mathcal{C}(m)$.

Step 2. This time we work with the dual form of hypothesis \mathcal{H} . Let \mathbf{B} be a random uniform $(k + 1) \times n$ matrix. Let \mathbf{s} be a binary vector of length $k + 1$ which is

- (i) either of the form $\mathbf{B}\mathbf{e}^\top$, where \mathbf{e} is a random vector of \mathbb{F}_2^n of weight t ,

(ii) or uniformly random in \mathbb{F}_2^{k+1} .

Hypothesis $\mathcal{H}(n, n - (k + 1), t)$ tells us that it should not be possible to distinguish whether \mathbf{s} was sampled from distribution (i) or from distribution (ii). Now we call \mathbf{A} the $k \times n$ matrix made up from the first k rows of \mathbf{B} and call \mathbf{y}' the last row of \mathbf{B} . Let \mathbf{s}_1 be the column vector made up of the first k coordinates of \mathbf{s} and let \mathbf{s}_2 be the last symbol (bit) of \mathbf{s} . We now choose m to be a random *bit* and define the pair

$$\mathcal{C}(m) = (\mathbf{s}_1, m + \mathbf{s}_2).$$

We then feed algorithm \mathcal{D} the matrix \mathbf{A} , the vector \mathbf{y}' and $\mathcal{C}(m)$, claiming $(\mathbf{A}', \mathbf{y}')$ to be an encryption key and $\mathcal{C}(m)$ to be a encryption of m . When we are in case (i), then we have $\mathcal{C}(m) = (\mathbf{A}\mathbf{e}^\top, m + \langle \mathbf{y}', \mathbf{e} \rangle)$ which must therefore be accepted as a valid encryption of m according to step 1. Accordingly, the decryption algorithm \mathcal{D} should succeed in returning m (with a good enough success probability). Now we apply hypothesis $\mathcal{H}(n, n - (k + 1), t)$ to claim that it should also succeed when we are in case (ii), otherwise we are able to distinguish between cases (i) and (ii). But this is absurd, because in case (ii) the quantity $\mathcal{C}(m)$ is just random noise together with a one-time pad of m (the quantity $m + \mathbf{s}_2$ is random uniform and independent of m and \mathbf{s}_1 , and it is impossible to extract from it any information on m).

7.3. Making the scheme more efficient. Suppose we now wish to encrypt two bits, so that now the plaintext is $m = (m_1, m_2)$ chosen from $\mathcal{M} = \{0, 1\}^2$. Of course, we could encrypt each bit m_1, m_2 separately with the previous scheme: but we can make encryption more efficient. Let the public key be made up of the $k \times n$ random matrix \mathbf{A} , as before, but now we add to it two vectors \mathbf{y}_1 and \mathbf{y}_2 instead of one, where each \mathbf{y}_i is sampled as before, namely $\mathbf{y}_1 = \mathbf{x}_1\mathbf{A} + \boldsymbol{\varepsilon}_1$ and $\mathbf{y}_2 = \mathbf{x}_2\mathbf{A} + \boldsymbol{\varepsilon}_2$, with $\mathbf{x}_1, \mathbf{x}_2$ independent and uniform in \mathbb{F}_2^k and $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2$ independent and uniform among vectors of weight t in \mathbb{F}_2^n .

Encryption. A random vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t is chosen and we set:

$$(11) \quad \mathcal{C}(m) = \left(\mathcal{C}_1 = \mathbf{A}\mathbf{e}^\top, \mathcal{C}_2(m) = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} + \begin{bmatrix} \langle \mathbf{y}_1, \mathbf{e} \rangle \\ \langle \mathbf{y}_2, \mathbf{e} \rangle \end{bmatrix} \right)$$

Decryption. It works pretty much as before, we compute

$$\mathcal{C}_2 + \begin{bmatrix} \mathbf{x}_1 \mathcal{C}_1 \\ \mathbf{x}_2 \mathcal{C}_1 \end{bmatrix}$$

which equals with probability close to 1.

Security argument. Very little is changed.

Step 1. We argue that the decryption algorithm \mathcal{D} should keep working when we change the public-key elements $\mathbf{y}_1, \mathbf{y}_2$ by $\mathbf{y}'_1, \mathbf{y}'_2$ which are two independent uniform vectors in the whole space \mathbb{F}_2^n . This is done through two substeps. First we invoke

hypothesis $\mathcal{H}(n, k, t)$ to claim that the decryption algorithm \mathcal{D} should continue to work when we just replace \mathbf{y}_1 by \mathbf{y}'_1 . Indeed, if we have an instance \mathbf{A}, \mathbf{y}_1 of the distinguishing problem $\mathcal{H}(n, k, t)$ we can add to it a vector $\mathbf{y}_2 = \mathbf{x}_2 \mathbf{A} + \boldsymbol{\varepsilon}_2$ that we choose of the right form, i.e. $\boldsymbol{\varepsilon}_2$ of weight t , and then create an encryption $\mathcal{C}(m)$ of some message $m \in \mathcal{M}$ through the formula (11): we then feed to \mathcal{D} the input $\mathbf{A}, \mathbf{y}_1, \mathbf{y}_2$. If \mathbf{y}_1 is sampled from distribution (ii), then $\mathcal{C}(m)$ is a valid encryption of m and the decryption algorithm \mathcal{D} is required to work and return m with a reasonable success probability. Decryption should work with the same (up to a negligible difference) success probability when \mathbf{y}_1 is sampled from distribution (i) instead, otherwise we would have an efficient distinguisher for (i) and (ii), contradicting hypothesis $\mathcal{H}(n, k, d)$.

Once we have established that the decryption algorithm \mathcal{D} continues to work with \mathbf{y}_1 replaced by \mathbf{y}'_1 , we replace \mathbf{y}_2 by \mathbf{y}'_2 and claim that this should again make no difference to the output of \mathcal{D} . Indeed, now \mathbf{A} together with \mathbf{y}'_1 make up a uniform random $(k+1) \times n$ matrix \mathbf{A}' , and together with either \mathbf{y}_2 or \mathbf{y}'_2 we have an instance the distinguishing problem $\mathcal{H}(n, k+1, t)$. If algorithm \mathcal{D} behaved differently with \mathbf{y}'_2 than with \mathbf{y}_2 , we would therefore violate hypothesis $\mathcal{H}(n, k+1, t)$.

Step 2. This is almost the same as in the base case. The matrix \mathbf{B} and the vector \mathbf{s} are chosen as before, except that \mathbf{B} is a $(k+2) \times n$ matrix, and $\mathbf{s} \in \mathbb{F}_2^{k+2}$. We now let \mathbf{A} be the $k \times n$ matrix made up of the first k rows of \mathbf{B} and define \mathbf{y}'_1 and \mathbf{y}'_2 to be the last two rows of \mathbf{B} . Similarly, we let \mathbf{s}_1 and \mathbf{s}_2 be the vectors made up of the first k coordinates of \mathbf{s} and the last two coordinates respectively. We choose a random $m \in \mathcal{M}$ and create $\mathcal{C}(m) = (\mathbf{s}_1, m + \mathbf{s}_2)$ as before (except that m and \mathbf{s}_2 are now 2-bit strings instead of 1-bit strings). We then feed the decryption algorithm the matrix \mathbf{A} , the vectors $\mathbf{y}'_1, \mathbf{y}'_2$ and $\mathcal{C}(m)$. We then claim

- (1) that the decryption algorithm \mathcal{D} should work when \mathbf{s} is of the form $\mathbf{B}\mathbf{e}^\top$ because then the input to \mathcal{D} is a valid public-key and a valid encryption of a plaintext m (according to step 1),
- (2) and that it should work just as well when \mathbf{s} is uniformly chosen in \mathbb{F}_2^{k+2} , otherwise we would have a distinguishing algorithm for cases (i) and (ii) in hypothesis $\mathcal{H}(n, n - (k+2), t)$.

But then we have a contradiction, because we end up with a decryption algorithm that should somehow be able to decrypt a one-time pad. \square

Compared to the base case, we have essentially divided by 2 the size of the cryptogram without hurting the security proof. This is because the size of the cryptogram is dominated by its first term, $\mathcal{C}_1 = \mathbf{A}\mathbf{e}^\top$, the term \mathcal{C}_2 being negligible, regardless of whether it is 1-bit long or 2-bit long.

It is natural to want to keep improving the scheme by encrypting more than 1 or 2 bits simultaneously.

7.4. Even greater efficiency. Let us now define the public key to be the random uniform $k \times n$ matrix \mathbf{A} , together with an $\ell \times n$ matrix \mathbf{Y} , consisting of ℓ rows $\mathbf{y}_1, \dots, \mathbf{y}_\ell$, each of which is constructed as $\mathbf{y}_i = \mathbf{x}_i \mathbf{A} + \boldsymbol{\varepsilon}_i$ where \mathbf{x}_i is uniform random in \mathbb{F}_2^k and $\boldsymbol{\varepsilon}$ is uniform random of weight t . More precisely,

$$\mathbf{Y} = \mathbf{X}\mathbf{A} + \mathbf{E}$$

with \mathbf{X} a uniform random $\ell \times k$ matrix and \mathbf{E} an $\ell \times n$ matrix whose rows are independent and uniform of weight t .

It is natural to define the plaintext message space by $\mathcal{M} = \mathbb{F}_2^\ell$ and to encrypt $m \in \mathcal{M}$ as

$$\mathcal{C}(m) = (\mathcal{C}_1 = \mathbf{A}\mathbf{e}^\top, \mathcal{C}_2(m) = m + \mathbf{Y}\mathbf{e}^\top).$$

To decrypt we compute $\mathcal{C}_2(m) + \mathbf{X}\mathcal{C}_1$ which is easily seen to be equal to

$$m + \mathbf{E}\mathbf{e}^\top.$$

However, if we want ℓ to scale as a constant fraction of n , then we cannot expect $\mathbf{E}\mathbf{e}^\top$ to be zero with a probability close to 1, for any non-zero value of t . To deal with this we introduce an auxiliary code C_0 , of length ℓ and dimension k_0 and that comes with an efficient decoding algorithm that allows us to decode a sufficiently large number ρ of errors. We now adapt the encryption scheme as follows:

Public key. The matrices \mathbf{A}, \mathbf{Y} together with a $k_0 \times \ell$ generator matrix \mathbf{G}_0 for the code C_0 , typically chosen in systematic form $G_0 = [I_{k_0} | M_0]$.

Secret key. The matrix \mathbf{X} .

Encryption. The plaintext message space is defined as $\mathcal{M} = \mathbb{F}_2^{k_0}$. A vector $\mathbf{e} \in \mathbb{F}_2^n$ is chosen uniformly among vectors of weight t , and the encrypted message is constructed as:

$$\mathcal{C}(m) = (\mathcal{C}_1 = \mathbf{A}\mathbf{e}^\top, \mathcal{C}_2(m) = m\mathbf{G}_0 + \mathbf{Y}\mathbf{e}^\top).$$

Decryption. Compute $\mathcal{C}_2(m) + \mathbf{X}\mathcal{C}_1 = m\mathbf{G}_0 + \mathbf{E}\mathbf{e}^\top$. The parameter t and the error-correcting radius ρ of the decoding algorithm for C_0 are to be chosen so that the typical weight of the vector $\mathbf{E}\mathbf{e}^\top$ is not more than ρ . Decoding the vector $m\mathbf{G}_0 + \mathbf{E}\mathbf{e}^\top$, we therefore obtain $m\mathbf{G}_0$, from which we recover m by keeping its first k_0 coordinates (when \mathbf{G}_0 has been chosen in systematic form).

Security argument. The argument works as before: a first step consists of replacing one by one the rows $\mathbf{y}_1, \dots, \mathbf{y}_\ell$ of \mathbf{Y} by uniform vectors $\mathbf{y}'_1, \dots, \mathbf{y}'_\ell$ and claiming that the decryption algorithm \mathcal{D} must work just as well, otherwise we contradict hypothesis $\mathcal{H}(n, k, t)$. The second step consists of choosing a random plaintext message m , and giving the decryption algorithm the new public key \mathbf{A}, \mathbf{Y}' (where the rows of \mathbf{Y}' are now the \mathbf{y}'_i 's), together with a pair $(\mathbf{s}_1, m\mathbf{G}_0 + \mathbf{s}_2)$. We argue that the decryption algorithm should return m when $\mathbf{s}_1, \mathbf{s}_2$ are chosen as $\mathbf{s}_1 = \mathbf{A}\mathbf{e}^\top$ and $\mathbf{s}_2 = \mathbf{Y}'\mathbf{e}^\top$ with $\mathbf{e} \in \mathbb{F}_2^n$ of weight t , since this is a properly formed

encryption of m . We also claim that the decryption algorithm should work just as well when $\mathbf{s}_1, \mathbf{s}_2$ are independent uniform random vectors since we would otherwise be able to distinguish distributions (i) and (ii) in hypothesis $\mathcal{H}(n, n - (k + \ell), t)$ (dual form). But this is again a contradiction since we would have a decryption algorithm that decrypts a one-time pad. \square

Remark 7.1.

- The security of the scheme relies on the difficulty of decoding two random codes. The random code of dimension k generated by the rows of \mathbf{A} (associated to hypothesis $\mathcal{H}(n, k, t)$) and the random code of dimension $n - (k + \ell)$ defined by the parity-check matrix $\begin{pmatrix} \mathbf{A} \\ \mathbf{V} \end{pmatrix}$ (associated to hypothesis $\mathcal{H}(n, n - (k + \ell), t)$).
- When choosing the parameters k and ℓ , one should make sure that $k + \ell \ll n$, otherwise the random code in hypothesis $\mathcal{H}(n, n - (k + \ell), t)$ has very small dimension, and it becomes feasible to decode it by exhaustively checking all codewords. Equivalently, it becomes feasible to solve the constrained linear system $\begin{pmatrix} \mathbf{A} \\ \mathbf{V} \end{pmatrix} \mathbf{e}^\top = \mathbf{s}$ and the second step of the security proof breaks down. More generally, hypothesis $\mathcal{H}(n, k, t)$ can only be trusted when k is well separated from 0 or n .
- The code C_0 plays no cryptographic role in the scheme and must not be confused with the random codes discussed above. The decoding algorithm for C_0 can be public, the security proof does not assume any secret knowledge for C_0 .