Summer 25

Information Set Decoding

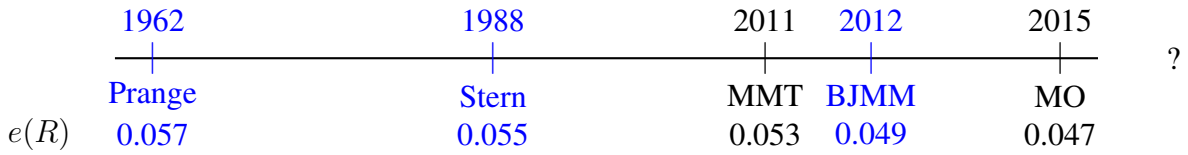
Lecturer: Violetta Weger

These lecture notes were prepared for the ENCODE Summer School, July 2025. Any comments or corrections can be sent to `violetta.weger@tum.de` The distribution of the notes is of course allowed.

Overview The main task of Information Set Decoding (ISD) is to decode a random linear code: Given a random $n-k \times n$ matrix H and vector s of length $n-k$, find a vector e of length n and Hamming weight up to t, such that $eH^\top = s$.

Information set decoding dates back to the algorithm of Prange in 1962 - which for coding theory is pretty old.

Until today, over 40 ISD algorithms have emerged. Clearly, we will not cover all of those algorithms. The major ones ([25, 26, 21, 6, 22]) are depicted in the timeline below. The ones we cover (if time permits) are in blue, where $e(R)$ denotes their asymptotic cost, i.e., their cost is $2^{e(R)n+o(n)}$ for $q = 2$.



Hence, we can also see on this timeline how little the cost of these algorithms has decreased. We left out some improvements by polynomial factors [20, 19] or algorithms achieving the same cost with different methods [15, 12].

We can clearly see the renewed interest in these algorithms since 2010, when the threat of quantum computers gave a boost to research in code-based cryptography in general.

This lecture is meant to give an introduction and a foundation on this research direction, hopefully helping you understand the main idea and the used techniques.

The main motivation to study these algorithms lies in their use in code-based cryptography, where at the heart of code-based cryptosystems we find the problem of decoding a random linear code.

This problem is known to be NP-hard and so we expect any solver, in particular information set decoders to have an exponential cost.

However, any improvement (especially targeting specific structures, such as quasi-cyclic, extension fields or regular weight distribution) could have a great impact on the proposed systems - or more precisely on their proposed parameters.

We will conclude this lecture with some of the *many* open questions which still remain after over 60 years of information set decoding.

Material: Most of the content of these lecture notes is collected from the seminal papers

- "Decoding random linear codes in." by Alexander May, Alexander Meurer, and Enrico Thomae. In International Conference on the Theory and Application of Cryptology and Information Security, 2011. [21]
- "Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding." By Anja Becker, Antoine Joux, Alexander May and Alexander Meurer. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2012. [6]
- "On computing nearest neighbors with applications to decoding of binary linear codes." By Alexander May and Ilya Ozerov. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2015. [22]

and the nice overviews provided in

- The Ph.D. dissertation of Alexander Meurer: "A coding-theoretic approach to cryptanalysis." Ruhr university Bochum, 2013. [24]
- The survey on code-based cryptography by me, Niklas Gassner and Joachim Rosenthal. "A survey on code-based cryptography.", 2022. [27]

Hence, if this introductory lecture got you interested, I highly recommend reading these references.

Since we cannot include everything, we will assume a certain background, most importantly: finite fields and coding theory. For more details on these topics, you may consult the appendix: Appendix 7.1, respectively Appendix 7.2.

Contents

Notation

Throughout these lecture notes, we will make use of the following notation

- For a set S we denote by $|S|$ its cardinality and by S^C its complement.
- Id_n denotes the identity matrix of size n.
- $\mathrm{GL}_q(n)$ denotes the general linear group of degree n in $\mathbb{F}_q$, i.e., all invertible matrices in $\mathbb{F}_q^{n\times n}$.
- For a matrix M we write $\mathrm{rk}(M)$ to denote its rank, $\det(M)$ to denote its determinant and by $M^\top$ we denote its transpose.
- For a function f we denote by $\ker(f)$ its kernel and by $\mathrm{im}(f)$ its image.
- For a vector $v \in \mathbb{F}_q^k$ and $i \in \{1,\dots,k\}$ we denote by v_i the ith entry of the vector v. For a subset $S \subset \{1,\dots,k\}$ of size s, $v_S \in \mathbb{F}_q^s$ denotes the vector consisting of the entries of v indexed by S.
- Similarly for a matrix: for a matrix $M \in \mathbb{F}_q^{k\times n}$ and $i \in \{1,\dots,k\}, j \in \{1,\dots,n\}$ we denote by $M_{i,j}$ the entry of M in the ith row and jth column. For a subset $S \subset \{1,\dots,n\}$ of size s, $M_S \in \mathbb{F}_q^{k\times s}$ denotes the matrix consisting of the columns of M indexed by S.
- For a matrix $M \in \mathbb{F}_q^{k\times n}$, we denote by $\langle M\rangle \subseteq \mathbb{F}_q^n$ the span of the rows of M, that is $\mathrm{im}(M) = \langle M\rangle$.

1 The Decoding Problem

Usually, we are interested in using algebraic codes for reliable communication, storage, network coding, information retrieval etc.

In all these scenarios, we want the codes to have an efficient decoder, meaning that if we are given a corrupted codeword, we can recover the codeword in polynomial time.

However, if we have no information on the algebraic structure of the code, or even try to decode a random code, a new question arises: how hard is it to decode in general?

This will be the main task of this chapter: *generic decoding*.

The main problem of this chapter, is called *Decoding Problem (DP).*

Problem 1.1 (Decoding Problem)**.** *Let $\mathbb{F}_q$ be a finite field and $k \leq n$ be positive integers. Given $G \in \mathbb{F}_q^{k\times n}$, $r \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$, find $e \in \mathbb{F}_q^n$ with $wt_H(e) \leq t$ and $r - e \in \langle G \rangle$.*

Clearly, generic decoders are algorithms that solve the DP.

1.1 Interlude: Code-based Cryptography

Such generic decoders have a great impact for cryptography: coding theory can also be used in cryptography. This intersection is called *code-based cryptography* and is as old as the RSA cryptosystem (i.e., from 1978 [23]).

On a very high level, the idea of the McEliece public-key encryption scheme is to use a code with an efficient decoder (he suggested Goppa codes) as a secret key, and to publish an equivalent code, where the structure of the secret code is hidden (and thus also the necessary information for the decoder). That is, we take as secret key $G \in \mathbb{F}_q^{k\times n}$, a generator matrix for the secret code, and publish $G' = SGP$ for some $S \in \mathrm{GL}_q(k), P$ a permutation matrix, and the error-correction capability t.

Anyone can then encrypt a message $m \in \mathbb{F}_q^k$, by computing a corrupted codeword:

$$c = mG' + e,$$

where $\mathrm{wt}_H(e) \leq t$.

The person with the secret key, i.e., S, P, and the decoding algorithm of $\mathcal{C}$, can then compute

$$cP^\top = mSG + eP^\top = m'G + e',$$

and since $\mathrm{wt}_H(eP^\top) = \mathrm{wt}_H(e') = \mathrm{wt}_H(e) \leq t$, the decoder for $\mathcal{C} = \langle G \rangle$, will return $m' = mS$. Finally by multiplying with S^{-1}, we recover the message m.

An eavesdropper has only access to the public key, i.e., G' and the corrupt codeword c. Thus their main task, is to decode a (seemingly) random code.

Clearly, in this scenario, the public code is not random: it is still an equivalent code to a Goppa code. By now, we also have new code-based cryptosystems, where we employ actually random codes.

For more details on public-key encryption, have a look at the Appendix 7.5.

Table 1: McEliece Cryptosystem

ALICE		BOB
KEY GENERATION		
Choose $G \in \mathbb{F}_q^{k\times n}$, a generator matrix of a code with efficient decoder $\mathcal{D}$ and error-correction capability t. Choose randomly $S \in \mathrm{GL}_k(q)$ and an $n \times n$ permutation matrix P. Compute $G' = SGP$. The public key is given by $\mathcal{P} = (t, G')$ and the secret key is $\mathcal{S} = (G, S, P, \mathcal{D})$		
	$\xrightarrow{\mathcal{P}}$	
		ENCRYPTION
		Choose a message $m \in \mathbb{F}_q^k$ and a random $e \in \mathbb{F}_q^n$ with $\mathrm{wt}_H(e) \leq t$ Ciphertext: $c = mG' + e$
	$\xleftarrow{c}$	
DECRYPTION		
Compute $\mathcal{D}(cP^{-1}) = \mathcal{D}(mSG + eP^{-1}) = mS$, and recover the message as $m = (mS)S^{-1}$		

1.2 Related Problems

Note that the DP is formulated through the generator matrix and we can get an equivalent formulation using the parity-check matrix: the *Syndrome Decoding Problem (SDP)*.

Problem 1.2 (Syndrome Decoding Problem)**.** *Let $\mathbb{F}_q$ be a finite field and $k \leq n$ be positive integers. Given $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$ and $t \in \mathbb{N}$,find $e \in \mathbb{F}_q^n$ such that $wt_H(e) \leq t$ and $eH^\top = s$.*

These two problems are also equivalent to the *Codeword Finding Problem (CFP)*:

Problem 1.3 (Codeword Finding Problem)**.** *Let $\mathbb{F}_q$ be a finite field and $k \leq n$ be positive integers. Let $k \leq n$ be positive integers. Given $H \in \mathbb{F}_q^{(n-k)\times n}$ and $t \in \mathbb{N}$, find $c \in \mathbb{F}_q^n$ such that $wt_H(c) \leq t$ and $cH^\top = 0$.*

Theorem 1.4. *The DP, SDP and CFP are equivalent.*

Proof. Let us start with showing that the DP and SDP are equivalent. For this we start with an instance of DP, i.e., G, r, t. We can then transform this instance to an instance of the SDP. In fact,

we can bring G into systematic form, that is $\begin{pmatrix} \mathrm{Id}_k & A \end{pmatrix}$ and immediately get a parity-check matrix for the same code $H = \begin{pmatrix} -A^\top & \mathrm{Id}_{n-k} \end{pmatrix}$.

We can then multiply H to the received vector $r = mG + e$, getting the syndrome

$$s = rH^\top = eH^\top.$$

Hence, if we can solve the SDP on the instance H, s, t, that is we find e, we have also solved the DP.

On the other hand, given an instance of SDP, i.e., H, s, t, we can find an instance of DP: we bring H into systematic form $\begin{pmatrix} B & \mathrm{Id}_{n-k} \end{pmatrix}$ and read of a generator matrix $G = \begin{pmatrix} \mathrm{Id}_k & -B^\top \end{pmatrix}$ for the same code.

We can now solve

$$xH^\top = s \tag{1}$$

for some unknown $x \in \mathbb{F}_q^n$ and since this is a linear system of $n - k$ equations in n unknowns, we get $N = q^k$ possible solutions: $x_1, \ldots, x_N$. Note that for each of the q^k codewords $c_1, \ldots, c_N$, we have that $c_i + e$ is a possible solution to (1). Thus, each of the q^k solutions x_i correspond to some $c_i + e$.

Hence, any of the solutions x_i can be used as received vector r and we have recovered an instance of DP, as G, r, t. Hence, solving DP, i.e., finding e, also solves the SDP instance.

Finally, we show that the DP and SDP are also equivalent to CFP. For this, we recall that the error-correction capability t is set as $t = \lfloor \frac{d-1}{2} \rfloor$, where d denotes the minimum distance of $\langle G \rangle$.

Given an instance of DP, i.e., G, r, t we can add r as a row to the generator matrix, getting

$$G' = \begin{pmatrix} G \\ r \end{pmatrix}.$$

Note that the code generated by G' is also generated by

$$\begin{pmatrix} G \\ e \end{pmatrix},$$

as $r = mG + e$. The new code of dimension $k + 1$ has now as lowest weight codeword λe of weight t for some $\lambda \in \mathbb{F}_q^\times$.

In fact, if there would exist some codeword $a \in \langle G' \rangle$ of weight t, which is not of the form $a = \lambda e$, then a must also involve codewords of $\langle G \rangle$, that is $a = c + be$, for $c \in \langle G \rangle \setminus \{0\}$ and some $b \in \mathbb{F}_q$.

Since we know $\mathrm{wt}_H(e) \leq t$, we must have that $c = a - be \in \langle G \rangle \setminus \{0\}$ has weight

$$\mathrm{wt}_H(c) = \mathrm{wt}_H(a - be) \leq \mathrm{wt}_H(a) + \mathrm{wt}_H(-be) \leq 2t < d,$$

a contradiction to the minimum distance of $\langle G \rangle$ being d.

Hence, we can compute the corresponding parity-check matrix H' of $\langle G' \rangle$ and solving the CFP on the instance H', t we recover the solution e to the DP instance.

On the other hand, given an instance H, t of the CFP, we can define an instance of SDP, by taking the same parity-check matrix and setting the syndrome $s = 0$. Thus, a solver for SDP, searching for a weight t vector e with $eH^\top = 0$ also solves the CFP instance. □

Thus, we may choose which of the three problems we wish to solve with our generic decoder. For this lecture, we will stick to the SDP.

1.3 The Hardness

Our first question, "how hard is it to decode in general?" has already been solved: The SDP has been proven to be NP-hard [7, 5], meaning it is one of the hardest problems in mathematics.

The problem is further also in NP. This means, if we are given a candidate solution e' we can easily (in polynomial time) check whether it is actually a solution, i.e., if $\mathrm{wt}_H(e') \leq t$ and $eH^\top = s$. Thus, the SDP is a NP-complete problem, which makes it a perfect candidate for cryptography.

For more details on complexity classes, you may consult Appendix 7.3.

We prove the NP-hardness of the SDP through a polynomial-time reduction. For this, we choose the 3-dimensional matching (3DM) problem, which is a well-known NP-hard problem.

Problem 1.5 (3-Dimensional Matching (3DM) Problem)**.** *Let T be a finite set and $U \subseteq T \times T \times T$. Given U, T, decide if there exists a set $W \subseteq U$ such that $\mid W \mid = \mid T \mid$ and no two elements of W agree in any coordinate.*

Proposition 1.6. *The SDP is NP-complete.*

Proof. We prove the NP-completeness by a polynomial-time reduction from the 3DM problem. For this, we start with a random instance of 3DM with T of size t, and $U \subseteq T \times T \times T$ of size u. Let us denote the elements in $T = \{b_1, \ldots, b_t\}$ and in $U = \{a_1, \ldots, a_u\}$. From this we build the matrix $H^\top \in \mathbb{F}_q^{u \times 3t}$, as follows:

- for $j \in \{1, \ldots, t\}$, we set $h_{i,j} = 1$ if $a_i[1] = b_j$ and $h_{i,j} = 0$ else,
- for $j \in \{t+1, \ldots, 2t\}$, we set $h_{i,j} = 1$ if $a_i[2] = b_j$ and $h_{i,j} = 0$ else,
- for $j \in \{2t+1, \ldots, 3t\}$, we set $h_{i,j} = 1$ if $a_i[3] = b_j$ and $h_{i,j} = 0$ else.

With this construction, we have that each row of $H^\top$ corresponds to an element in U, and has weight 3. Let us set the syndrome s as the all-one vector of length $3t$. Assume that we can solve the SDP on the instances H, s and t in polynomial time. Let us consider two cases.

Case 1: First, assume that the SDP solver returns as answer 'yes', i.e., there exists an $e \in \mathbb{F}_q^u$, of weight less than or equal to t and such that $eH^\top = s$.

- We first observe that we must have $\mathrm{wt}_H(e) = \mid \mathrm{supp}_H(e) \mid = t$. For this note that each row of $H^\top$ adds at most 3 non-zero entries to s. Therefore, we need to add at least t rows to get s, i.e., $\mid \mathrm{supp}_H(e) \mid \geq t$ and hence $\mathrm{wt}_H(e) \geq t$. As we also have $\mathrm{wt}_H(e) \leq t$ by hypothesis, this implies that $\mathrm{wt}_H(e) = \mid \mathrm{supp}_H(e) \mid = t$.

- Secondly, we observe that the weight t solution must be a binary vector. For this we note that the matrix $H^\top$ has binary entries and has constant row weight three, and since $\mid \mathrm{supp}_H(e) \mid = t$, the supports of the t rows of $H^\top$ that sum up to the all-one vector have to be disjoint. Therefore, we get that the j-th equation from the system of equations $eH^\top = s$ is of the form $e_i h_{i,j} = 1$ for some $i \in \mathrm{supp}_H(e)$. Since $h_{i,j} = 1$, we have $e_i = 1$.

Recall from above that the rows of $H^\top$ correspond to the elements of U. The t rows corresponding to the support of e are now a solution W to the 3DM problem. This follows from the fact that the t rows have disjoint supports and add up to the all-one vector, which implies that each element of T appears exactly once in each coordinate of the elements of W.

Case 2: Now assume that the SDP solver returns as answer 'no', i.e., there exists no $e \in \mathbb{F}_q^u$ of weight at most t such that $eH^\top = s$. This response is now also the correct response for the 3DM problem. In fact, if there exists $W \subseteq U$ of size t such that all coordinates of its elements are distinct, then t rows of $H^\top$ should add up to the all one vector, which in turn means the existence of a vector $e \in \{0,1\}^u$ of weight t such that $eH^\top = s$.

Thus, if such a polynomial time solver exists, we can also solve the 3DM problem in polynomial time. □

Example 1.7. *Let us consider* $T = \{A, B, C, D\}$ *and*

$$U = \{(D, A, B), (C, B, A), (D, A, B), (B, C, D), (C, D, A), (A, D, A), (A, B, C)\}.$$

Then the above construction would yield

$$H^\top = \left[\begin{array}{cccc|cccc|cccc} 0&0&0&1&1&0&0&0&0&1&0&0 \\ 0&0&1&0&0&1&0&0&1&0&0&0 \\ 0&0&0&1&1&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0&1 \\ 0&0&1&0&0&0&0&1&1&0&0&0 \\ 1&0&0&0&0&0&0&1&1&0&0&0 \\ 1&0&0&0&0&1&0&0&0&0&1&0 \end{array}\right].$$

A solution to $eH^\top = (1, \ldots, 1)$ *would be* $e = (1, 0, 0, 1, 1, 0, 1)$ *which corresponds to*

$$W = \{(D, A, B), (B, C, D), (C, D, A), (A, B, C)\}.$$

2 What is a Random Code?

Now that the task is clear: we have to solve one of the hardest problems in mathematics, we can work out our assumptions on the instance.

These assumptions are motivated from the application in code-based cryptography.

Recall that the SDP is given $H \in \mathbb{F}_q^{(n-k)\times n}, s \in \mathbb{F}_q^{n-k}, t \in \mathbb{N}$ and searches for a vector $e \in \mathbb{F}_q^n$ with

1. $eH^\top = s$ (called syndrome equation)
2. and $\mathrm{wt}_H(e) \leq t$ (called weight constraint).

2.1 Syndrome Equation

First of all, what do we mean with *random* code?

If we were to choose $\mathcal{C}$ uniform at random from the space

$$\{\mathcal{C} \subset \mathbb{F}_q^n \mid \dim(\mathcal{C}) = k\},$$

of size

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \prod_{i=0}^{k-1} \frac{(q^n - q^i)}{(q^k - q^i)}$$

then each of the $\mathcal{C}$ has multiple parity-check matrices. In fact, for a parity-check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ also SH is a parity-check matrix, where $S \in \mathrm{GL}_q(n-k)$.

Instead, we say $\mathcal{C}$ is random, if $\mathcal{C}$ is generated by a matrix $G \in \mathbb{F}_q^{k\times n}$ chosen uniform at random of full rank. Equivalently, we can choose $H \in \mathbb{F}_q^{(n-k)\times n}$ uniform at random of full rank and set $\mathcal{C} = \ker(H^\top)$.

To ensure the full rank condition, we usually set $H = \begin{pmatrix} \mathrm{Id}_{n-k} & A \end{pmatrix}$, for some A chosen uniform at random from $\mathbb{F}_q^{(n-k)\times k}$.

This is (more or less) legitimate, as

$$q^{(n-k)k} \leq \begin{bmatrix} n \\ k \end{bmatrix}_q \leq 4q^{(n-k)k}.$$

Note that since H is chosen uniform at random, also the syndrome is chosen uniform at random from $\mathbb{F}_q^{n-k}$.

Lemma 2.1. *Let $H \in \mathbb{F}_q^{(n-k)\times n}$. Then for any $x \in \mathbb{F}_q^n \setminus \{0\}$, the syndrome $s = xH^\top$ is chosen uniform at random from $\mathbb{F}_q^{n-k}$.*

Proof. Let us denote the columns of H by $h_i \in \mathbb{F}_q^{n-k}$. We note that for each $i \in \{1, \ldots, n-k\}$ we have

$$s_i = \langle h_i, x \rangle = \sum_{j=1}^{n} x_j h_{j,i}$$

is also chosen uniform at random. □

What is the probability that a random $x \in \mathbb{F}_q^n$ is such that $xH^\top = s$?

Lemma 2.2. *Let $H \in \mathbb{F}_q^{(n-k)\times n}$ and $s \in \mathbb{F}_q^{n-k}$. The probability of a $x \in \mathbb{F}_q^n$ chosen uniform at random to be such that $xH^\top = s$ is given by $q^{-(n-k)}$.*

Proof. We first note that for a fixed $H \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$ the map $x \mapsto xH^\top$ is not injective. We can have $x \neq x'$ with $s = xH^\top = x'H^\top$. This implies that $x - x' \in \ker(H^\top)$. Since $\ker(H^\top)$ has dimension k, for a fixed $x \in \mathbb{F}_q^n$ there exist q^k many x_i such that $x - x_i \in \ker(H^\top)$.

Since this is true for any syndrome $s \in \mathbb{F}_q^{n-k}$, we have that each $s \in \mathbb{F}_q^{n-k}$ is the image of q^k many $x \in \mathbb{F}_q^n$. Thus, for a random $x \in \mathbb{F}_q^n$ the probability to be such that $xH^\top = s$ is $\frac{q^k}{q^n} = q^{-(n-k)}$. □

2.2 Weight Constraint

For the weight constraint, we want to restrict ourselves to weights t up to the correction capability of the code, that is $t \leq \lfloor \frac{d-1}{2} \rfloor$, where d denotes the minimum distance of the code $\ker(H^\top)$, and (due to its use in cryptography) we will assume that a solution e exists. Thus, it is also the unique solution.

Exercise 2.3. *Show that if $e, e' \in \mathbb{F}_q^n$ are both solutions to the SDP with $t \leq \lfloor \frac{d-1}{2} \rfloor$, then $e = e'$.*

We cannot choose any t and assume the problem is still hard; in fact, we will see that if $\tau = t/n \in [(1-R)\frac{q-1}{q}, R + (1-R)\frac{q-1}{q}]$ we have a polynomial time solver.

This is not a problem, as we know how the minimum distance of a random code behaves (for large n).

Let us denote by $V_H(r, n, q)$ is the size (or volume) of a ball and $A(d, n, q)$ the largest size of *any* code (also non-linear), in $\mathbb{F}_q^n$ with minimum distance at least d.

The Gilbert-Varshamov (GV) bound provides a lower bound on $A(d, n, q)$.

Theorem 2.4 (Gilbert-Varshamov Bound)**.** *Let q be a prime power and n, d be positive integers. Then,*

$$A(d, n, q) \geq \frac{q^n}{V_H(d-1, n, q)}.$$

Note that this bound is an existence bound. It should be read as follows:

There exists a (possibly non-linear) code which has size larger than $\frac{q^n}{V_H(d-1,n,q)}$.

This does *not* imply that a given code should have size larger, equal or smaller than $\frac{q^n}{V_H(d-1,n,q)}$. In fact, many codes, e.g. RS codes have size smaller than $\frac{q^n}{V_H(d-1,n,q)}$ and there exist codes which have a larger size.

A similar argument, also tells us the number of solutions to a SDP instance.

Lemma 2.5. *The number of solutions of a SDP instance $H \in \mathbb{F}_q^{(n-k)\times n}, s \in \mathbb{F}_q^{n-k}, t$ is given by*

$$\frac{|\{x \in \mathbb{F}_q^n \mid wt_H(x) \leq t\}|}{|\{x \in \mathbb{F}_q^n \mid xH^\top = s\}|} = \frac{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i}{q^{n-k}}.$$

Exercise 2.6. *Prove Lemma 2.5.*

It turns out that random codes attain the asymptotic GV bound with high probability for n growing.

Before we can give the asymptotic version of the GV bound, recall the definition of the entropy function:

Definition 2.7 (Entropy Function)**.** For a positive integer $q \geq 2$ the q-ary entropy function is defined as follows:

$$\begin{aligned} H_q : [0,1] &\to \mathbb{R}, \\ x &\to x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x). \end{aligned}$$

We want to find

$$\lim_{n\to\infty} \frac{1}{n}\log_q(V_H(rn,n,q)),$$

for some $r \in [0, 1-1/q]$.

Lemma 2.8. *Let $q \geq 2$ and n a positive integer and $x \in [0, 1-1/q]$. Then,*

$$V_H(xn,n,q) \leq q^{H_q(x)n}.$$

Proof. We have that

$$\begin{aligned} \frac{V_H(xn,n,q)}{q^{H_q(x)n}} &= \frac{\sum_{i=0}^{xn}\binom{n}{i}(q-1)^i}{(q-1)^{xn}x^{-xn}(1-x)^{-(1-x)n}} \\ &= \sum_{i=0}^{xn}\binom{n}{i}(q-1)^i(q-1)^{-xn}x^{xn}(1-x)^{(1-x)n} \\ &= \sum_{i=0}^{xn}\binom{n}{i}(q-1)^i(1-x)^n\left(\frac{x}{(q-1)(1-x)}\right)^{xn}. \end{aligned}$$

Since $x \leq 1 - 1/q$, we get that $x/(q-1) \leq 1/q \leq 1 - x$ and hence $\frac{x}{(q-1)(1-x)} < 1$ and by decreasing the power, we increase the formula. Thus,

$$\begin{aligned}\frac{V_H(xn,n,q)}{q^{H_q(x)n}} &\leq \sum_{i=0}^{xn} \binom{n}{i}(q-1)^i(1-x)^n \left(\frac{x}{(q-1)(1-x)}\right)^i \\ &= \sum_{i=0}^{xn} \binom{n}{i}(1-x)^{n-i}x^i \\ &\leq \sum_{i=0}^{n} \binom{n}{i}(1-x)^{n-i}x^i = 1,\end{aligned}$$

where we used the binomial theorem in the last equality. □

We can also give a lower bound, using Sterling's formula. For this, recall that

$$m! = \sqrt{2\pi m}\left(\frac{m}{e}\right)^m (1 + o(1))$$

and hence

$$\binom{n}{xn} \geq \left(\frac{1}{x}\right)^{xn} \left(\frac{1}{1-x}\right)^{(1-x)n} \exp(-o(n)) = 2^{H_q(x)n - o(n)}.$$

From this we can follow that

$$V_H(xn,n,q) \geq \binom{n}{xn}(q-1)^{xn} \geq q^{H_q(x)n-o(n)}.$$

Corollary 2.9. *Let $q \geq 2$ and n a positive integer and $x \in [0, 1 - 1/q]$. Then,*

$$\lim_{n\to\infty} \frac{1}{n} \log_q(V_H(xn,n,q)) = H_q(x).$$

We see k, d as a function in n and set

$$\delta = \lim_{n\to\infty} \frac{d(n)}{n} \in [0, 1 - 1/q], \quad R = \lim_{n\to\infty} \frac{1}{n} \log_q A(\delta n, n, q)$$

to be the relative minimum distance and the asymptotic rate.

The asymptotic Hamming bound then states

Corollary 2.10. *Let $\mathcal{C}$ be an $[n,k,d]_q$ linear code and assume that k and d are functions in n, while q is fixed. Then,*

$$R \leq 1 - H_q(\delta/2) + o(1).$$

We can formulate the asymptotic Gilbert-Varshamov bound as the existence of a infinite family of codes as

Theorem 2.11 (The Asymptotic Gilbert-Varshamov Bound)**.** *For every prime power q and $\delta \in [0, 1 - 1/q]$ there exists an infinite family $\mathcal{C}$ of codes with rate*

$$R(\delta) \geq 1 - H_q(\delta).$$

Theorem 2.12. *Let q be a prime power, $\delta \in [0, 1 - 1/q)$ and $0 < \varepsilon$ and n a positive integer. Set $k \leq n(1 - H_q(\delta) - \varepsilon)$ and let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a random code of dimension k. Then, with high probability, $\mathcal{C}$ has minimum Hamming distance at least δn, for n growing.*

Proof. We want to show that

$$\mathbb{P}(d_H(\mathcal{C}) > \delta n) \geq 1 - q^{-\varepsilon n}.$$

We first choose $G \in \mathbb{F}_q^{k \times n}$ uniform at random of rank k and then bound the counter probability, that is

$$\mathbb{P}(d_H(\mathcal{C}) \leq \delta n)$$

which is given by the probability that there exists a non-zero codeword of weight at most δn. Since G is uniform at random, also any non-zero codeword $mG \in \mathbb{F}_q^n \setminus \{0\}$ is uniform at random.

We note that for a random non-zero codeword the probability of having weight at most δn can be bounded as

$$\mathbb{P}(\mathrm{wt}_H(mG) \leq \delta n) = \frac{V_H(\delta n, n, q)}{q^n - 1} \leq q^{n(H_q(\delta)-1)}.$$

Thus, using a union bound, we get

$$\begin{aligned}
\mathbb{P}(d_H(\mathcal{C}) \leq \delta n) &= \mathbb{P}(\exists m \in \mathbb{F}_q^k \setminus \{0\} : \mathrm{wt}_H(mG) \leq \delta n) \\
&\leq \sum_{m \in \mathbb{F}_q^k \setminus \{0\}} \mathbb{P}(\mathrm{wt}_H(mG) \leq \delta n) \\
&\leq (q^k - 1) q^{n(H_q(\delta)-1)} \\
&\leq q^{n(1-H_q(\delta)-\varepsilon)+n(H_q(\delta)-1)} = q^{-\varepsilon n}.
\end{aligned}$$

Hence $\mathbb{P}(d_H(\mathcal{C}) > \delta n) \geq 1 - q^{-\varepsilon n}$ and $\lim_{n \to \infty} \mathbb{P}(d_H(\mathcal{C}) > \delta n) = 1$. □

Thus, for large n, we now know that random codes attain with high probability the Gilbert-Varshamov bound, that is we may set

$$d_H(\mathcal{C}) = \max \left\{ r \;\middle|\; \sum_{i=0}^{r-1} \binom{n}{i} (q-1)^i < q^{n-k} \right\}.$$

Or equivalently, we can set $\delta = H_q^{-1}(1 - R)$, and $t = n\delta/2$.

3 First Solvers: Brute Force and Prange

The two conditions on e of the SDP are not compatible:

1. the parity-check equation $eH^\top = s$, is a linear constraint, while
2. the weight constraint $\mathrm{wt}_H(e) \leq t$ is non-linear.

One condition alone is clearly not hard to solve: the first one is a linear system with $n-k$ equations and n unknowns, for which we can find a solution in polynomial time. We can also simply list all vectors of weight t.

However, it is very unlikely that any solution for one of the conditions will also satisfy the other. In particular, since we assumed that in their intersection, we only have a unique solution.

The first try we could have at solving the DP is straightforward: solve only one of the conditions and check for the other. These are the two brute-force algorithms.

3.1 Brute Force

The first Brute-Force Algorithm, simply solves the syndrome equation and checks whether a found solution has the target weight.

Algorithm 1 Brute-Force Decoding 1

Input: $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $t \in \mathbb{N}$.
Output: $e \in \mathbb{F}_q^n$ with $eH^\top = s$ and $\mathrm{wt}_H(e) \leq t$.

1: Find the solution set $\mathcal{L}$ to the linear system $xH^\top = s$.
2: **for** $x \in \mathcal{L}$ **do**
3: **if** $\mathrm{wt}_H(x) \leq t$ **then**
4: Return x

Proposition 3.1. *The Brute-Force Algorithm 1 has a cost in $\mathcal{O}(q^k)$.*

Proof. Recall that there exist q^k many vectors with syndrome s. This follows easily from the observation that the solution set $\mathcal{L}$ is expected to have size q^k. Out of all these vectors only one has weight $\leq t$, thus the cost of such brute-force algorithm is in

$$\mathcal{O}(|\mathcal{L}|) = \mathcal{O}(q^k).$$

□

Similarly, we can go through the vectors of weight t and check if the syndrome equations are satisfied. For this we denote by

$$B_H(t, n, q) = \{x \in \mathbb{F}_q^n \mid \mathrm{wt}_H(x) \leq t\}$$

the Hamming ball of radius t.

Algorithm 2 Brute-Force Decoding 2

Input: $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $t \in \mathbb{N}$.
Output: $e \in \mathbb{F}_q^n$ with $eH^\top = s$ and $\mathrm{wt}_H(e) \leq t$.

1: Build the list $B_H(t, n, q)$ of all vectors of weight $\leq t$.
2: **for** $x \in B_H(t, n, q)$ **do**
3: **if** $xH^\top = s$ **then**
4: Return x

Proposition 3.2. *The Brute-Force Algorithm 2 has a cost in* $\mathcal{O}\left(\sum_{i=0}^{t} \binom{n}{i}(q-1)^i\right)$.

Proof. Note that

$$B_H(t, n, q) = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$$

and since there exists a unique solution to the SDP, we are expected to go through all elements of $B_H(t, n, q)$. □

3.2 Information Set Decoding Idea

There are of course a more clever algorithms to solve SDP than brute-forcing, but we note that their costs will remain exponential.

The main idea is to use the information sets. These types of algorithms have been initialized by Prange in 1962 [25], and are called *Information Set Decoding (ISD)* algorithms.

Recall the definition and the properties of information sets:
An $[n, k]_q$ linear code can be completely defined by having access only to (correctly chosen) k positions. The following concept characterizes such defining sets.

Definition 3.3 (Information Set)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n, k]_q$ linear code. Then, a set $I \subset \{1, \ldots, n\}$ of size k is called an *information set* of $\mathcal{C}$ if

$$| \mathcal{C} | = | \mathcal{C}_I | .$$

Exercise 3.4. *How many information sets can an* $[n, k]_q$ *linear code have at most?*

Proposition 3.5. *Let* $\mathcal{C}$ *be an* $[n, k]_q$ *linear code,* I *an information set and let* G *be a generator matrix . The matrix* G_I *is an invertible matrix.*

Proposition 3.6. *Let* $\mathcal{C}$ *be an* $[n, k]_q$ *linear code,* I *an information set and* H *a parity-check matrix. If* $I^C := \{1, \ldots, n\} \setminus I$ *is the complement set of* I*, then,* H_{I^C} *is an invertible matrix.*

Finally, we can also compute the probability of a chosen set I to be an information set.

The probability for a random set $I \subset \{1, \ldots, n\}$ of size k to be an information set is large and ignored in the cost computations:

$$\frac{\prod_{i=0}^{k-1}(q^k - q^i)}{q^{k^2}} = \prod_{i=1}^{k}(1 - q^{-i}).$$

Note that if $k = Rn \to \infty$, we have $\prod_{i=1}^{k}(1 - q^{-i}) \to 1$.

As all ISD algorithms follow a similar structure, we will first introduce them on a high-level: We start by picking a set J of size $k + \ell$ containing an information set and assume a certain weight of e_J, say w, and thus impose e_{J^C} has the remaining weight $t - w$.

By doing so, we may solve a smaller problem than the initial SDP instance.

In fact, we can bring H into a *quasi-systematic form*, i.e.,

$$H' = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & A \\ 0 & B \end{pmatrix},$$

where $A \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}, B \in \mathbb{F}_q^{\ell\times(k+\ell)}$. We then also split the syndrome s' accordingly: $s' = \begin{pmatrix} s_1 & s_2 \end{pmatrix}$, where $s_1 \in \mathbb{F}_q^{n-k-\ell}$ and $s_2 \in \mathbb{F}_q^{\ell}$.

The parity-check equation $e'H'^{\top} = s'$ becomes two equations:

$$e_{J^C} + e_J A^{\top} = s_1, \tag{2}$$

$$e_J B^{\top} = s_2. \tag{3}$$

Note that if we find a solution $e_J \in \mathbb{F}_q^{k+\ell}$ of weight w, to (3), we can simply check if

$$\mathrm{wt}_H(e_{J^C}) = \mathrm{wt}_H(s_1 - e_J A^{\top}) = t - w.$$

Thus, we have reduced the initial SDP with instance (H, s, t) into a smaller SDP with instance (B, s_2, w).

The solution to the smaller instance is now not unique. In fact, using Lemma 2.5 we get

$$\frac{\sum_{i=0}^{w}\binom{k+\ell}{i}(q-1)^i}{q^{\ell}} > 1.$$

We may summarize the general idea of ISD as:

1. Find a set $J \subset \{1, \ldots, n\}$ of size $k + \ell$ containing an information set for $\mathcal{C}$.
2. Find an invertible matrix $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$, and a permutation matrix P, such that
$$UHP = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & A \\ 0 & B \end{pmatrix}.$$
3. Compute $s' = sU^\top$ and split it into s_1, s_2.
4. Find $e_J \in \mathbb{F}_q^{k+\ell}$ of weight w and such that $e_J B^\top = s_2$.
5. Check if $\mathrm{wt}_H(e_{J^C}) = \mathrm{wt}_H(s_1 - e_J A^\top) = t - w$.
6. If this is satisfied, output $e = (e_J, e_{J^C})P^\top$, if not start over with a new choice of J.

Note that for a fixed set J, the sought error vector e might not be such that e_J has weight w. Thus, the iteration above has to be repeated several times, and the final cost of such algorithm is given by the cost of one iteration times the expected number of required iterations.

On average, the number of iterations required is given by the reciprocal of the success probability of one iteration and this probability is completely determined by the assumed weight distribution.

Lemma 3.7. *Let $k \leq n, \ell \leq n - k$ and $w \leq t$ be positive integers. Let $e \in \mathbb{F}_q^n$ be of weight t. For a randomly chosen $J \subset \{1, \ldots, n\}$ of size $k + \ell$, the probability that $\mathrm{wt}_H(e_J) = w$ is given by*
$$\binom{t}{w}\binom{n-t}{k+\ell-w}\binom{n}{k+\ell}^{-1}.$$

We note that fixing e and going through all possible choices of J, is indeed what the algorithm tells us to do. However, to compute the success probability of one iteration, it is usually easier to go the other direction: fix a set J and compute the probability that e has the desired weight distribution.

Lemma 3.8. *Let $k \leq n$ and $w \leq t$ be positive integers. Let $J \subset \{1, \ldots, n\}$ be of size $k + \ell$. For a randomly chosen $e \in \mathbb{F}_q^n$ of weight t, the probability that $\mathrm{wt}_H(e_J) = w$ is given by*
$$\binom{k+\ell}{w}\binom{n-k-\ell}{t-w}\binom{n}{t}^{-1}.$$

Exercise 3.9. *Prove Lemma 3.7 and 3.7 and show that the two probabilities are the same.*

3.3 Prange's Algorithm

In Prange's algorithm we assume that there exists an information set I that is disjoint to the support of the error vector $\text{supp}_H(e)$, i.e.,

$$I \cap \text{supp}_H(e) = \emptyset.$$

Thus, in terms of our previous general algorithm for ISD, we set $w = \ell = 0$ and hence $J = I$ and $e_I = 0$.

To illustrate the algorithm, let us assume that the information set is $I = \{1, \ldots, k\}$. To bring the parity-check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ into systematic form, we multiply by an invertible matrix $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$. Since we assume that no errors occur in the information set, we have that $e = (0, e_{I^C})$ with $\text{wt}_H(e_{I^C}) = t$. We are in the following situation:

$$UHe^\top = \begin{pmatrix} \text{Id}_{n-k} & A \end{pmatrix} \begin{pmatrix} e_{I^C}^\top \\ 0^\top \end{pmatrix} = Us^\top,$$

for $A \in \mathbb{F}_q^{(n-k)\times k}$.

It follows that $e_{I^C} = sU^\top$ and hence we are only left with checking the weight of $s' = sU^\top$.

Algorithm 3 Prange's Algorithm

Input: $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $t \in \mathbb{N}$.
Output: $e \in \mathbb{F}_q^n$ with $eH^\top = s$ and $\text{wt}_H(e) = t$.

1: Choose an information set $I \subset \{1, ..., n\}$ of size k.
2: Compute $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$, such that

$$(UH)_I = A \text{ and } (UH)_{I^C} = \text{Id}_{n-k},$$

where $A \in \mathbb{F}_q^{(n-k)\times k}$.
3: Compute $s' = sU^\top$.
4: **if** $\text{wt}_H(s') = t$ **then**
5: Return e such that $e_I = 0$ and $e_{I^C} = s'$.
6: Start over with Step 1 and a new selection of I.

Theorem 3.10. *Prange's algorithm has a cost in*

$$\mathcal{O}\left(\binom{n-k}{t}^{-1}\binom{n}{t}\right).$$

binary operations.

Proof. One iteration of Algorithm 3 only consists of bringing H into systematic form and applying the same row operations on the syndrome; thus, the cost can be assumed equal to that of computing $U \begin{pmatrix} H & s^\top \end{pmatrix}$, i.e., $(n-k)^2(n+1)$ $\mathbb{F}_q$-operations or

$$(n-k)^2(n+1)(\lceil \log_2(q) \rceil + \lceil \log_2(q) \rceil^2)$$

binary operations.

The success probability is given by having chosen the correct weight distribution of e. In this case, we require that no errors happen in the chosen information set, hence the probability is given by

$$\binom{n-k}{t}\binom{n}{t}^{-1}.$$

Since the average number of iterations are then $\binom{n-k}{t}^{-1}\binom{n}{t}$ and

$$(n-k)^2(n+1)\left(\lceil \log_2(q) \rceil + \lceil \log_2(q) \rceil^2\right)\binom{n-k}{t}^{-1}\binom{n}{t} \in \mathcal{O}\left(\binom{n-k}{t}^{-1}\binom{n}{t}\right),$$

the Gaussian elimination part introduces only polynomial factors which we may ignore. □

Let us consider an example for Prange's algorithm.

Example 3.11. *Let us consider* $\mathbb{F}_5$ *and*

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 3 & 4 \end{pmatrix}, \qquad s = (2, 4, 1)$$

and we are looking for $e \in \mathbb{F}_5^3$ *with weight* $t = 1$.

We might start with the information set $I_1 = \{4, 5\}$ *as* H *is already in systematic form for* I_1. *That is the necessary* $U_1 = Id_3$, *however* $s' = s$ *does not have weight* $t = 1$.

Instead, the information set $I_2 = \{1, 2\}$ *leads to*

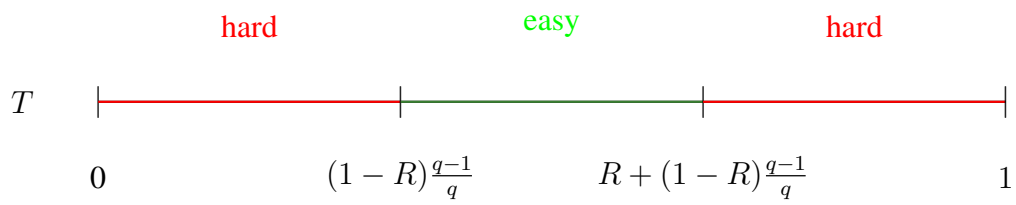
$$U_2 = \begin{pmatrix} 1 & 3 & 1 \\ 2 & 2 & 0 \\ 2 & 4 & 0 \end{pmatrix}, \quad \textit{i.e.,} \quad U_2H = \begin{pmatrix} 1 & 3 & 1 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 2 & 4 & 0 & 0 & 1 \end{pmatrix}.$$

Now we get $s' = sU_1^\top = (0, 2, 0)$ *has weight 1, and hence we found the error vector*

$$e = (0, 0, 0, 2, 0).$$

Recall that we claimed at the beginning that there is a region, where SDP is easy.

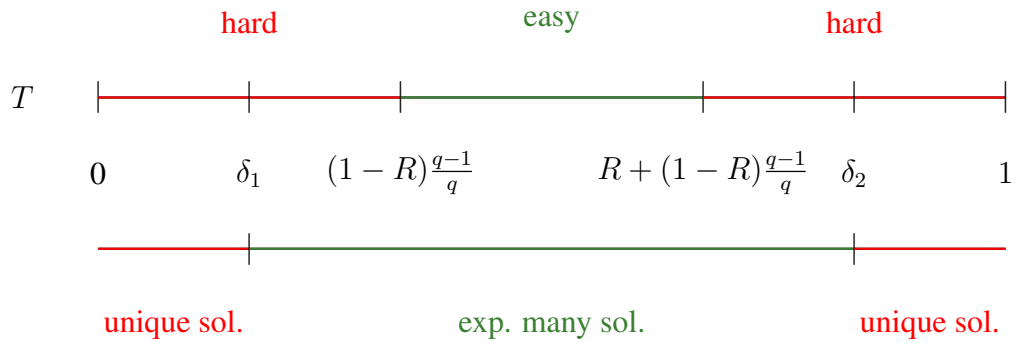
In fact, we have the following situation



This is not a problem, as we usually set $T = t/n$ to be half the minimum distance on the GV bound, that is $T = H_q^{-1}(1-R)/2$.

Using the same reasoning, we could also do "full distance decoding" and set $T = H_q^{-1}(1-R)$ as on average we still expect at most one solution. We also remark that the entropy function is not injective and there are two solutions to $H_q^{-1}(1-R)$, say $\delta_1 < 1/2$ and $\delta_2 > 1/2$.

Thus for our usual choice of T, we are in the "hard" region:



To see that for $t \in [(n-k)(q-1)/q, k+(n-k)(q-1)/q]$ decoding becomes polynomial time, we use Prange's idea.

Recall that we have

$$\left(\mathrm{Id}_{n-k} \quad A\right)(e_I, e_{I^C})^\top = s'^\top.$$

we set $e_I = 0$ and $e_{I^C} = s'$. Note that we expect a random $s' \in \mathbb{F}_q^{n-k}$ to have weight $\frac{q-1}{q}(n-k)$. Thus, with high probability, we get that $\mathrm{wt}_H(e_{I^C}) = t = \frac{q-1}{q}(n-k) = \mathrm{wt}_H(s')$.

We could also set $e_I \in \mathbb{F}_q^k$ to be anything else of weight $0 \leq \ell \leq k$ and set $\tilde{s} = s' - e_I A^\top$. Again we expect it to be of weight $\frac{q-1}{q}(n-k)$ and thus get that e_{I^C} has the remaining weight to get $\mathrm{wt}_H(e) = \ell + (n-k)\frac{q-1}{q}$.

4 The usual ISD: Stern's ISD

A few years later in 1988, Stern [26] proposed a meet-in-the-middle approach to solve for the smaller instance.

Recall that when we are given the instance $H \in \mathbb{F}_q^{(n-k)\times n}, s \in \mathbb{F}_q^{n-k}$ and t and are searching for $e \in \mathbb{F}_q^n$ such that $\mathrm{wt}_H(e) = t$ and $eH^\top = s$, we may first reduce it to a smaller instance, $e' \in \mathbb{F}_q^{k+\ell}$ with $\mathrm{wt}(e') = w$ and $e'H'^\top = s'$, where $H' \in \mathbb{F}_q^{\ell\times(k+\ell)}, s' \in \mathbb{F}_q^\ell$.

Stern proposes to split $e' = (e_1, e_2)$, where $e_i \in \mathbb{F}_q^{(k+\ell)/2}$ are of weight $w/2$ and similarly to split $H' = \begin{pmatrix} H_1 & H_2 \end{pmatrix}$, with $H_i \in \mathbb{F}_q^{\ell\times(k+\ell)/2}$. The syndrome equation $e'H\top = s'$ then becomes

$$\begin{pmatrix} H_1 & H_2 \end{pmatrix} \begin{pmatrix} e_1^\top \\ e_2^\top \end{pmatrix} = s'^\top$$

that is

$$e_1 H_1^\top + e_2 H_2^\top = s'.$$

If we denote (for the correct choice e) that $e_1 H_1^\top = s_1$ and $e_2 H_2^\top = s_2$, then it is again enough to find a pair (e_1, e_2) such that $s_1 + s_2 = s'$. Thus we simply set $s_1 = s' - e_2 H_2^\top$.

We may then build two lists

$$\begin{aligned}\mathcal{L}_1 &= \{(e_1 H_1^\top, e_1) \mid e_1 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}_H(e_1) = w/2\},\\ \mathcal{L}_2 &= \{(s' - e_2 H_2^\top, e_2) \mid e_2 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}_H(e_2) = w/2\}.\end{aligned}$$

If we find a collision, that is $((a, e_1), (a, e_2)) \in \mathcal{L}_1 \times \mathcal{L}_2$, we know that $e_1 H_1^\top = s' - e_2 H_2^\top$ and hence $(e_1, e_2)H'^\top = s'$.

Theorem 4.1. *Stern's algorithm has a cost in*

$$\mathcal{O}\left(\binom{(k+\ell)/2}{w/2}^{-2}\binom{n-k-\ell}{t-w}^{-1}\binom{n}{t}\left(\binom{(k+\ell)/2}{w/2}(q-1)^{w/2} + \binom{(k+\ell)/2}{w/2}^2(q-1)^{w-\ell}\right)\right).$$

Proof. We start with the cost of one iteration. Again, the computation of UH is only polynomial in n and thus negligible. On the other hand, the construction of the lists $\mathcal{L}_i$ costs

$$|\mathcal{L}_i| = \binom{(k+\ell)/2}{w/2}(q-1)^{w/2}.$$

To go through $\mathcal{L}_1 \times \mathcal{L}_2$ would usually cost $|\mathcal{L}_i|^2$, but since we are only interested in collisions, i.e., when $s' - e_2 H_2^\top = e_1 H_1^\top \in \mathbb{F}_q^\ell$, we can multiply $|\mathcal{L}_i|^2$ with the probability of having a collision, that is $q^{-\ell}$.

Algorithm 4 Stern's Algorithm

Input: $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $w < t, \ell < n-k$.
Output: $e \in \mathbb{F}_q^n$ with $eH^\top = s$ and $\mathrm{wt}_H(e) = t$.

1: Choose a set $J \subset \{1, ..., n\}$ of size $k + \ell$.
2: Compute $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$, such that

$$(UH)_J = \begin{pmatrix} A \\ H' \end{pmatrix}, \ (UH)_{J^C} = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} \\ 0 \end{pmatrix},$$

where $A \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$ and $H' \in \mathbb{F}_q^{\ell\times(k+\ell)}$.
3: Split $H' = (H_1, H_2)$, with $H_i \in \mathbb{F}_q^{\ell\times(k+\ell)/2}$.
4: Compute $sU^\top = \begin{pmatrix} \tilde{s} & s' \end{pmatrix}$, where $\tilde{s} \in \mathbb{F}_q^{n-k-\ell}$ and $s' \in \mathbb{F}_q^{\ell}$.
5: Compute the sets

$$\mathcal{L}_1 = \{(e_1 H_1^\top, e_1) \mid e_1 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}_H(e_1) = w/2\},$$
$$\mathcal{L}_2 = \{(s' - e_2 H_2^\top, e_2) \mid e_2 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}_H(e_2) = w/2\}.$$

6: **for** $(a, e_1) \in \mathcal{L}_1$ **do**
7: **for** $(a, e_2) \in \mathcal{L}_2$ **do**
8: **if** $\mathrm{wt}_H(\tilde{s} - (e_1, e_2)A^\top) = t - w$ **then**
9: Return e such that $e_J = (e_1, e_2)$, $e_{J^C} = \tilde{s} - (e_1, e_2)A^\top$.
10: Start over with Step 1 and a new selection of J.

We get that the cost of one iteration is in

$$\mathcal{O}\left(\binom{(k+\ell)/2}{w/2}(q-1)^{w/2} + \binom{(k+\ell)/2}{w/2}^2 (q-1)^{w-\ell}\right).$$

For the success probability of one iteration, we need to compute

$$\frac{|\{e \in \mathbb{F}_q^n \mid e_J = (e_1, e_2), \mathrm{wt}_H(e_i) = w/2, \mathrm{wt}_H(e_{J^C}) = t - w\}|}{|\{e \in \mathbb{F}_q^n \mid \mathrm{wt}_H(e) = t\}},$$

which is given by

$$\binom{(k+\ell)/2}{w/2}^2 \binom{n-k-\ell}{t-w} \binom{n}{t}^{-1}.$$

Thus the overall cost of Stern's algorithm is in

$$\mathcal{O}\left(\binom{(k+\ell)/2}{w/2}^{-2} \binom{n-k-\ell}{t-w}^{-1} \binom{n}{t} \left(\binom{(k+\ell)/2}{w/2}(q-1)^{w/2} + \binom{(k+\ell)/2}{w/2}^2 (q-1)^{w-\ell}\right)\right).$$

□

Example 4.2. *Let us consider again* $\mathbb{F}_5, n = 10, k = 4, t = 3, w = 2$, *and* $\ell = 2$.

Let

$$H = \left(\begin{array}{ccccc} & 1 & 2 & 3 & 1 \\ & 2 & 4 & 1 & 2 \\ & 3 & 3 & 2 & 4 \\ \mathit{Id}_6 & 1 & 2 & 1 & 3 \\ & 4 & 1 & 1 & 2 \\ & 3 & 3 & 4 & 1 \end{array}\right), \qquad s = (1, 0, 3, 1, 4, 4).$$

If we set $J = \{5, 6, 7, 8, 9, 10\}$ *then this clearly contains the information set* $I = \{7, 8, 9, 10\}$*, for which* H *is already in systematic form.*

Thus, we get

$$H' = \begin{pmatrix} 1 & 0 & 4 & 1 & 1 & 2 \\ 0 & 1 & 3 & 3 & 4 & 1 \end{pmatrix}, \quad s' = (4, 4).$$

We can then split H' *into*

$$H_1 = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \end{pmatrix}, H_2 = \begin{pmatrix} 1 & 1 & 2 \\ 3 & 4 & 1 \end{pmatrix}.$$

We build the lists

$$\begin{aligned} \mathcal{L}_1 &= \{(e_1 H_1^\top, e_1) \mid e_1 \in \mathbb{F}_5^3, \mathit{wt}_H(e_1) = 1\} \\ &= \{((\lambda, 0), (\lambda, 0, 0)), ((0, \lambda), (0, \lambda, 0)), ((4\lambda, 3\lambda), (0, 0, \lambda)) \mid \lambda \in \mathbb{F}_5^\times\}, \\ \mathcal{L}_2 &= \{(s' - e_2 H_2^\top, e_2) \mid e_2 \in \mathbb{F}_5^3, \mathit{wt}_H(e_2) = 1\} \\ &= \{((4 - \lambda, 4 - 3\lambda), (\lambda, 0, 0)), ((4 - \lambda, 4 - 4\lambda), (0, \lambda, 0)), ((4 - 2\lambda, 4 - \lambda), (0, 0, \lambda)) \mid \lambda \in \mathbb{F}_5^\times\}. \end{aligned}$$

Both lists have size $12 = \binom{3}{1}(5 - 1)^1 = 3 \cdot 4$.

We then search for collisions among the two lists, we find

$$\begin{aligned}
&((1,0),(1,0,0)) \in \mathcal{L}_1, ((1,0),(3,0,0)) \in \mathcal{L}_2,\\
&((3,0),(3,0,0)) \in \mathcal{L}_1, ((3,0),(0,1,0)) \in \mathcal{L}_2,\\
&((1,0),(1,0,0)) \in \mathcal{L}_1, ((1,0),(0,0,4)) \in \mathcal{L}_2,\\
&((0,1),(0,1,0)) \in \mathcal{L}_1, ((0,1),(4,0,0)) \in \mathcal{L}_2,\\
&((0,3),(0,3,0)) \in \mathcal{L}_1, ((0,3),(0,4,0)) \in \mathcal{L}_2,\\
&((0,2),(0,2,0)) \in \mathcal{L}_1, ((0,2),(0,0,2)) \in \mathcal{L}_2,\\
&((3,1),(0,0,2)) \in \mathcal{L}_1, ((3,1),(1,0,0)) \in \mathcal{L}_2,\\
&((1,2),(0,0,4)) \in \mathcal{L}_1, ((1,2),(0,3,0)) \in \mathcal{L}_2,\\
&((3,1),(0,0,2)) \in \mathcal{L}_1, ((3,1),(0,0,3)) \in \mathcal{L}_2.
\end{aligned}$$

Which are more than the expected $|\mathcal{L}_i|^2 5^{-2} = 5.76$ *collisions.*

For each of the candidate $e' = (e_1, e_2)$ *we compute*

$$x = \tilde{s} - e' A^\top = (1,0,3,1) - (e_1, e_2) \begin{pmatrix} 0 & 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & 2 & 4 & 1 & 2 \\ 0 & 0 & 3 & 3 & 2 & 4 \\ 0 & 0 & 1 & 2 & 1 & 3 \end{pmatrix}^\top$$

and check if it has the remaining weight $t - w = 1$.

For $e' = (e_1, e_2) = (1,0,0,3,0,0)$ *we get* $x = (0,3,4,0)$ *which is not of weight* 1, *for* $e' = (3,0,0,0,1,0)$ *we get* $x = (3,4,1,0)$ *and we continue until the very last collision, where* $e' = (0,0,2,0,0,3)$ *and we get* $x = (1,0,0,0)$.

Hence, we set $e = (x, e') = (1,0,0,0,0,0,2,0,0,3)$ *which is of weight* $t = 3$ *and such that* $eH^\top = s$.

Note that Stern's algorithm is always at least as fast as Prange, as it recovers Prange by setting $w = \ell = 0$.

4.1 Asymptotic Cost

An important aspect of ISD algorithms (apart from the cost) is their asymptotic cost. The idea of the asymptotic cost is that we are interested in the exponent $e(R,q)$ such that for large n the cost of the algorithm is given by $q^{(e(R,q)+o(1))n}$. This is crucial in order to compare different algorithms.

We consider codes of large length n, and consider the dimension and the error correction capacity as functions in n, for which we define

$$\begin{aligned}
\lim_{n\to\infty} t(n)/n &= T,\\
\lim_{n\to\infty} k(n)/n &= R.
\end{aligned}$$

Recall that $H_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$ and due to the GV bound we can set $T = \delta/2$, where $\delta = H_q^{-1}(1-R)$. If $c(n,k,t,q)$ denotes the cost of an algorithm, for example Prange's algorithm, then we are now interested in

$$e(R,q) = \lim_{n\to\infty} \frac{1}{n}\log_q(c(n,k,t,q)).$$

For this we often use Stirlings formula, that is if

$$\lim_{n\to\infty} a(n)/n = A,$$
$$\lim_{n\to\infty} b(n)/n = B$$

then

$$\lim_{n\to\infty} \frac{1}{n}\log_q\binom{a}{b} = A\log_q(A) - B\log_q(B) - (A-B)\log_q(A-B).$$

Theorem 4.3. *The asymptotic cost of Prange's algorithm is $q^{(e(q,R)+o(1))n}$, where*

$$e(q,R) = -(1-T)\log_q(1-T) - (1-R)\log_q(1-R) + (1-R-T)\log_q(1-R-T),$$

where $T = H_q^{-1}(1-R)/2$.

Proof. Recall that the cost of Prange's algorithm is given by

$$c(n,k,t,q) = \binom{n-k}{t}^{-1}\binom{n}{t}.$$

Using Stirling's formula, we get that

$$\begin{aligned}
&\lim_{n\to\infty} \frac{1}{n}\log_q\left(\binom{n-k}{t}^{-1}\binom{n}{t}\right) = \\
&-((1-R)\log_q(1-R) - T\log_q(T) - (1-R-T)\log_q(1-R-T)) \\
&+1\log_q(1) - T\log_q(T) - (1-T)\log_q(1-T) \\
&= -(1-T)\log_q(1-T) - (1-R)\log_q(1-R) + (1-R-T)\log_q(1-R-T).
\end{aligned}$$

Finally, due to the GV we have that $T = H_q^{-1}(1-R)/2$. □

Exercise 4.4. *Prove that the asymptotic cost of Prange is equal to*

$$H_q(T) - (1-R)H_q(T/(1-R)).$$

Let us also compute the asymptotic cost of Stern. Since we have internal parameters ℓ, w we first need to set

$$\lim_{n\to\infty} \ell(n)/n = L,$$
$$\lim_{n\to\infty} w(n)/n = W.$$

Theorem 4.5. *The asymptotic cost of Stern's algorithm is $q^{(e(q,R)+o(1))n}$, where*

$$e(q,R) = \min_{L,W}\left\{-2A - B + C + \max\left\{A + \frac{W}{2}\log_q(q-1), 2A + (W-L)\log_q(q-1)\right\}\right\},$$

where

$$\begin{aligned}
A =& \frac{R+L}{2}\log_q\left(\frac{R+L}{2}\right) - \frac{W}{2}\log_q\left(\frac{W}{2}\right) - \frac{R+L-W}{2}\log_q\left(\frac{R+L-W}{2}\right),\\
B =& (1-R-L)\log_q(1-R-L) - (T-W)\log_q(T-W)\\
&- (1-R-L-T+W)\log_q(1-R-L-T+W),\\
C =& -T\log_q(T) - (1-T)\log_q(1-T).
\end{aligned}$$

Proof. Recall that the cost of Stern's algorithm is given by

$$\begin{aligned}
c(n,k,t,q) =& \binom{(k+\ell)/2}{w/2}^{-2}\binom{n-k-\ell}{t-w}^{-1}\binom{n}{t}\\
&\cdot\left(\binom{(k+\ell)/2}{w/2}(q-1)^{w/2} + \binom{(k+\ell)/2}{w/2}^{2}(q-1)^{w-\ell}\right).
\end{aligned}$$

We start by computing

$$\begin{aligned}
A &= \lim_{n\to\infty}\frac{1}{n}\log_q\left(\binom{(k+\ell)/2}{w/2}\right)\\
&= \frac{R+L}{2}\log_q\left(\frac{R+L}{2}\right) - \frac{W}{2}\log_q\left(\frac{W}{2}\right) - \frac{R+L-W}{2}\log_q\left(\frac{R+L-W}{2}\right).
\end{aligned}$$

We then compute

$$\begin{aligned}
B &= \lim_{n\to\infty}\frac{1}{n}\log_q\left(\binom{n-k-\ell}{t-w}\right)\\
&= (1-R-L)\log_q(1-R-L) - (T-W)\log_q(T-W)\\
&\quad - (1-R-L-T+W)\log_q(1-R-L-T+W).
\end{aligned}$$

Finally,

$$C = \lim_{n\to\infty}\frac{1}{n}\log_q\left(\binom{n}{t}\right) = -T\log_q(T) - (1-T)\log_q(1-T).$$

Thus, we get that

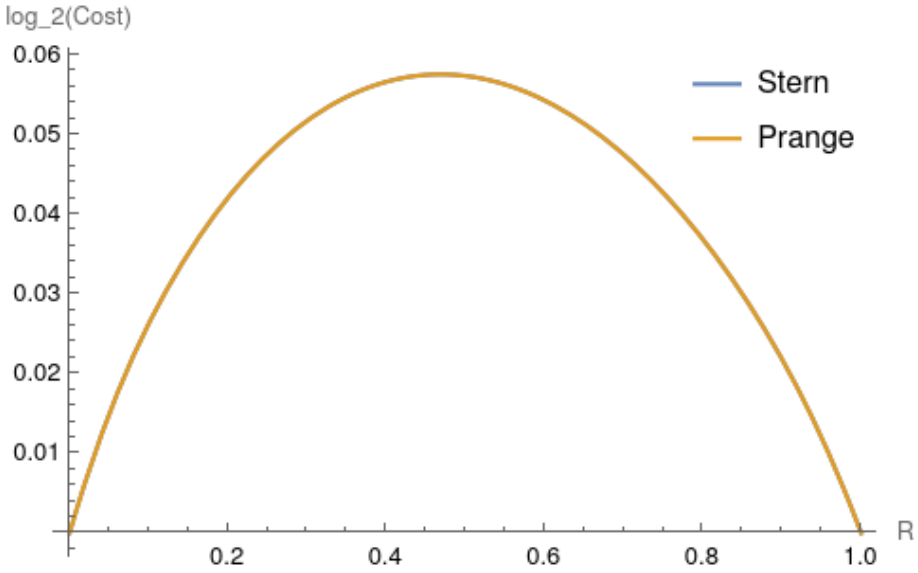
$$\begin{aligned}
&\lim_{n\to\infty}\frac{1}{n}\log_q\left(c(n,k,t,q)\right)\\
&= -2A - B + C + \max\left\{A + \frac{W}{2}\log_q(q-1), 2A + (W-L)\log_q(q-1)\right\}.
\end{aligned}$$

Since the algorithm will optimize the choices of ℓ, w with the restrictions

$$\ell < n - k - t + w, \qquad w < t.$$

□

We can then plot the cost for a fixed $q = 2$, as



Their difference is barely visible.

We often also give the maximal cost over all rates, that is

$$e^*(q) = \max\{e(R, q) \mid R \in [0, 1]\}.$$

We then get for $q = 2$ that

Algorithm	$e^*(q)$
Prange	0.05747
Stern	0.05563

Table 2: Comparison of Prange and Stern for $q = 2$.

5 More Advanced Algorithms: BJMM

I would love to show you the many ideas people have come up with to solve the SDP.

Due to time limitations, we will focus on one of the most interesting ones: BJMM [6]. For this algorithm, we will restrict ourselves to $q = 2$. A non-binary version has been proposed in [24].

5.1 The Idea: Representations

Recall that Stern's idea was to split the smaller instance of the error vector into two halves

$$e' = (e_1, e_2).$$

We could also think of this splitting as a sum:

$$e' = (e_1, 0) + (0, e_2) = x_1 + x_2.$$

Now instead of assuming that x_1 and x_2 have weight $w/2$ and thus *no* overlap in their support, we could generalize this to $e = x_1 + x_2$, where $\mathrm{wt}_H(x_i) = w/2 + \varepsilon$, that is we assume that

$$|\mathrm{supp}_H(x_1) \cap \mathrm{supp}_H(x_2)| = \varepsilon.$$

Since we are over the binary, any overlap in their support will cancel out when adding them.

Example 5.1. *Consider $e' = (1, 0, 1, 0, 1, 1)$, then we could choose*

$$x_1 = (1, 0, 1, 1, 0, 0), \quad x_2 = (0, 0, 0, 1, 1, 1)$$

and get

$$x_1 + x_2 = e'.$$

While before, we had a unique way of splitting e' into two halves (at least when the halves are fixed), now we have many different possibilities to write $e' = x_1 + x_2$.

Exercise 5.2. *Find all possible ways of writing $e' = (1, 0, 1, 0, 1, 1)$ as sum of two binary vectors of weight 3.*

That is: the main idea of BJMM is instead of only writing $e'_i = 0$ as $x_{1,i} + x_{2,i} = 0 + 0$, we make use of the binary structure as we could also have $e'_i = 1 = x_{i,1} + x_{2,i} = 1 + 1$.

This gives raise to the following definition.

Definition 5.3. Let $e' \in \mathbb{F}_2^{k+\ell}$ be of weight w. The number of pairs (x_1, x_2) with $x_i \in \mathbb{F}_2^{k+\ell}$ of weight $w/2 + \varepsilon$ and such that $x_1 + x_2 = e'$ is called the *number of representations.*

Lemma 5.4. *Let $e' \in \mathbb{F}_2^{k+\ell}$ be of weight w. The number of representations of weight $w/2 + \varepsilon$ is given by*

$$R(\varepsilon, w, \ell) = \binom{w}{w/2}\binom{k+\ell-w}{\varepsilon}.$$

Proof. We note that if $e'_i = 1$, we can only write it as $0 + 1$ or $1 + 0$, thus we have to split the support of e' into two halves, resulting in the first $\binom{w}{w/2}$. That is for $J_1 \cup J_2 = \mathrm{supp}_H(e')$ with size $w/2$ we set $x_{1,i} = 1, x_{2,i} = 0$ for all $i \in J_1$ and $x_{1,j} = 0, x_{2,j} = 1$ for all $j \in J_2$.

In the positions where $e'_i = 0$, which are $k + \ell - w$ many, we can now choose ε many positions, where we set $x_{1,i} = x_{2,i} = 1$. This gives us $\binom{k+\ell-w}{\varepsilon}$ many possibilities. □

To construct x_1, x_2 we could again build a list of all possible vectors

$$\mathcal{L} = \{x \in \mathbb{F}_q^{k+\ell} \mid \mathrm{wt}_H(x) = w/2 + \varepsilon\}$$

and merge them, that is construct the list

$$\mathcal{L}' = \{e' \in \mathbb{F}_q^{k+\ell} \mid e' = x_1 + x_2, x_i \in \mathcal{L}, x_1 H'^{\top} + x_2 H'^{\top} = s', \mathrm{wt}_H(e') = w\}$$

whenever the resulting sum e' has the desired weight w.

This, unfortunately, is not a good idea. We would get a collision search cost of

$$\frac{|\mathcal{L}|^2}{q^\ell},$$

which will optimize at setting $\varepsilon = 0$, that is: at Stern.

Additionally, we would clearly store several times the same e' in $\mathcal{L}'$, as we have seen that there are many pairs (x_1, x_2) adding up to the same e'. Thus, instead of stopping here, we have to go one level further.

5.2 The Subroutine: Stern

To construct $\mathcal{L}$, consisting of vectors x of length $k + \ell$ and weight $w/2 + \varepsilon$, we will use Stern's approach.

We can write $x = (y_1, y_2)$ where $y_i \in \mathbb{F}_q^{(k+\ell)/2}$ and $\mathrm{wt}_H(y_i) = w/4 + \varepsilon/2$.

Until now, we have not talked about the syndrome equation yet. Our new splitting

$$e' = x_1 + x_2$$

means that we also need

$$x_1 H'^{\top} + x_2 H'^{\top} = s'.$$

As we now start solving for x_1, x_2 independently, we need to fix a target syndrome for both of them, say $x_1 H'^{\top} = t_1$ and $x_2 H'^{\top} = t_2$ with $t_1 + t_2 = s'$.

Since any syndrome is equally likely, we may simply set $t_1 = s'$ and $t_2 = 0$.

Thus, we build our base lists:

$$\mathcal{B} = \{y \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}_H(y) = w/4 + \varepsilon/2\}$$

and start merging them à la Stern to the lists $\mathcal{L}_{x_1}, \mathcal{L}_{x_2}$.

We do need two different lists this time, as the merge happens differently: To construct $\mathcal{L}_{x_1}$, we take all $(y_1, y_2) \in \mathcal{B} \times \mathcal{B}$ which by construction are such that $x_1 = (y_1, y_2)$ has weight $w/2 + \varepsilon$, but we also require that

$$y_1 H_1^\top + y_2 H_2^\top = s'.$$

Instead for $\mathcal{L}_{x_2}$, we require that

$$y_1 H_1^\top + y_2 H_2^\top = 0.$$

Now clearly, it would not make sense for an algorithm to store several times the same $e' \in \mathcal{L}'$, as there exist $R(\varepsilon, w, \ell)$ many pairs $(x_1, x_2) \in \mathcal{L}_{x_1} \times \mathcal{L}_{x_2}$ which add up to the same e'.

Hence, we will not construct the whole set $\mathcal{L}_{x_i}$ but only a fraction of it: we will not ask $y_1 H_1^\top + y_2 H_2^\top$ to be equal to s', respectively 0 on *all* ℓ positions. Instead, we only require them to match the targets s', respectively 0 on

$$r = \log_2(R(\varepsilon, w, \ell))$$

many positions. This way, such collision search would result in lists of size

$$|\mathcal{L}_{x_i}| = \frac{|\mathcal{B}|^2}{q^r} = \frac{|\mathcal{B}|^2}{R},$$

which ensures that on average at least 1 representation (x_1, x_2) of e' will live in the resulting $\mathcal{L}_{x_1} \times \mathcal{L}_{x_2}$.

Let us summarize the steps of BJMM again:

1. Find a set $J \subset \{1, \ldots, n\}$ of size $k + \ell$ containing an information set for $\mathcal{C}$.
2. Find an invertible matrix $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$, and a permutation matrix P, such that
$$UHP = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & A \\ 0 & H' \end{pmatrix}.$$
3. Compute $sU^\top$ and split it into $\tilde{s}, s'$. Split $H' = (H_1, H_2)$.
4. Build the base list
$$\mathcal{B} = \{y \in \mathbb{F}_q^{(k+\ell)/2} \mathrm{wt}_H(y) = w/4 + \varepsilon/2\}.$$
5. Merge $\mathcal{L}_{x_1} = \mathcal{B} \times \mathcal{B}$ on the target s' for r many positions.
6. Merge $\mathcal{L}_{x_2} = \mathcal{B} \times \mathcal{B}$ on the target 0 for r many positions.
7. Merge $\mathcal{L}' = \mathcal{L}_{x_1} \times \mathcal{L}_{x_2}$.
8. For all $e' \in \mathcal{L}'$: check if $\mathrm{wt}_H(e_{J^C}) = \mathrm{wt}_H(\tilde{s} - e'A^\top) = t - w$.
9. If this is satisfied, output $e = (e', e_{J^C})P^\top$, if not start over with a new choice of J.

Note that the actual BJMM algorithm uses 3 levels and not two - even more complicated.

6 Summary and Open Questions

6.1 Summary

ISD algorithms aim at decoding a random linear code and are thus of exponential cost.

Their main idea is to make use of the information set of the code and thus to reduce the instance to a smaller SDP instance.

Their structure is as follows:

1. Find a set $J \subset \{1, \dots, n\}$ of size $k + \ell$ containing an information set for $\mathcal{C}$.
2. Find an invertible matrix $U \in \mathbb{F}_q^{(n-k)\times(n-k)}$, and a permutation matrix P, such that
$$UHP = \begin{pmatrix} \mathrm{Id}_{n-k-\ell} & A \\ 0 & B \end{pmatrix}.$$
3. Compute $s' = sU^\top$ and split it into s_1, s_2.
4. Find $e_J \in \mathbb{F}_q^{k+\ell}$ of weight w and such that $e_J B^\top = s_2$.
5. Check if $\mathrm{wt}_H(e_{J^C}) = \mathrm{wt}_H(s_1 - e_J A^\top) = t - w$.
6. If this is satisfied, output $e = (e_J, e_{J^C})P^\top$, if not start over with a new choice of J.

The cost of ISD algorithms is always given by

$$(\text{Cost of one iteration}) \cdot (\text{average number of iterations}).$$

ISD algorithms only differ in point 4.: How to solve the smaller instance.

- *Prange* solves it immediately by setting $e_J = 0$.
- *Stern* solves it by splitting $e_J = (e_1, e_2)$ and with a collision search.
- *BJMM* solves it by splitting $e_J = x_1 + x_2$ and using Stern as subroutine.

Clearly, there are many more improvements to the simple ideas of Prange and Stern, but most rely on the explained steps and have a similar cost analysis. In fact, over the last 60 years, the exponent $e^*(q)$ has only decreased from Prange's 0.05747 to 0.0473 [22].

Some of the newest methods are sieving [11, 16] or nearest-neighbor search [22, 14, 8].

While these are all nice ideas, in practice we (usually) only rely on Stern. This is due to the huge memory required in these improvements.

6.2 Open Problems

Although this research area is active and important for code-based cryptography there are many unsolved questions:

1. How to decode a (quasi-)cyclic code?
2. How to decode a q-ary code (faster)?
3. How to decode for large weights?
4. How to decode using quantum algorithms?
5. How to decode a regular error?
6. How to decode a restricted error?

Let us have a closer look at these open problems:

1. Note that the code-based cryptosystem to be standardized (by the U.S. authorities) is HQC [1], which relies on quasi-cyclic codes. None of the ISD algorithms (until now) is able to incorporate this additional structure to lower its cost.

2. New proposals use codes over $\mathbb{F}_{2^m}$ (such as the first version of SDitH [2]) and while the algorithms we have seen so far are also able to decode such q-ary codes, none of them use the additional structure of the extension field $\mathbb{F}_{2^m}$. In fact, all the q-ary algorithms are simply generalizations of the binary version.

We have several techniques to exploit an extension field structure, such as using the expansion map, the trace or the Frobenius, however, none of these ideas seem to improve the cost, leading to the conjecture that ISD over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$ (where p and 2^m are roughly of the same size) costs equally much.

For large q, the best decoder is the simple algorithm by Prange. In fact, all other decoders involve an enumeration step (that is build a list of vectors in $\mathbb{F}_q^{n'}$ of some weight w). As such lists have size $\binom{n'}{w}(q-1)^w$, the cost of such algorithms quickly grows too large. On the other hand, Prange's algorithm is oblivious of the underlying field.

3. The NIST round 1 proposal Wave [4] relies on a SDP instance where t is large. The usual way to deal with large weights, is too simplistic:

Let N be the number of solutions to the SDP (since we might not have a unique solution), then the cost of finding one solution is given by the usual cost, divided by N.

Additionally, the large weight decoders are again generalizations of the usual ISD algorithms, which were constructed for small weights.

4. There are several quantum ISD proposals [17, 18, 13]. However, they all use existing ISD algorithms and sprinkle some quantum algorithms (like Grover or quantum walks) on top.

5. The NIST round 2 proposal SDitH also relies on *regular* errors, meaning that the sought-after error vector $e = (e_1, \ldots, e_\ell)$, where $e_i \in \mathbb{F}_q^{n/\ell}$ and $\mathrm{wt}_H(e_i) = 1$. They adapt Stern's algorithm to this more structured problem, however, we could possibly do better than that.

6. In the NIST round 2 proposal CROSS [3], we drop the weight constraint, and instead ask for error vectors $e \in \mathbb{E}^n$, where

$$E = \{g^i \mid i \in \{0, \ldots, z-1\}\},$$

and $g \in \mathbb{F}_q^\star$ has order z. While we adapted Stern and BJMM to this scenario, there might be other attack strategies, which are -again- not simply adapting existing algorithms.

I hope you did learn something new or even better, you liked the topic.

If you got interested in ISD or you have some ideas on how to solve the open questions - do let me know!

7 Additional Material a.k.a. Appendix

7.1 Finite Fields

This section is covering the background on finite fields and their main properties, which might be useful for this lecture.

Summary

If you are already familiar with finite fields, here is a short summary of results, which prove to be useful for the course:

- For every prime p there is a unique finite field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ (up to isomorphism) of size p.
- For every prime p and positive integer m there is a unique finite field $\mathbb{F}_{p^m}$ (up to isomorphism) of size p^m. The subfield $\mathbb{F}_p$ is called the base field.
- The finite field $\mathbb{F}_{p^m}$ is a $\mathbb{F}_p$-linear vector space of dimension m over $\mathbb{F}_p$.
- The set $\mathbb{F}_{p^m}^\star = \mathbb{F}_{p^m} \setminus \{0\}$ is a cyclic multiplicative group.
- In any finite field $\mathbb{F}_{p^m}$ there exist $\varphi(p^m - 1)$ many primitive elements α, i.e.,
$$\mathbb{F}_{p^m}^\star = \{\alpha^0, \alpha^1, \ldots, \alpha^{p^m-2}\}.$$
- Any $x \in \mathbb{F}_{p^m}$ is such that $x^{p^m} = x$.
- For any $x \in \mathbb{F}_{p^m}$ with $x^p = x$, we have that $x \in \mathbb{F}_p$.
- For any prime p and positive integers m and ℓ such that $\ell \mid m$, the finite field $\mathbb{F}_{p^{m/\ell}}$ is a subfield of $\mathbb{F}_{p^m}$.
- For any finite field $\mathbb{F}_{p^m}$ the characteristic is p, i.e., for any $x \in \mathbb{F}_{p^m}$ we have $px = 0$.
- For any finite field $\mathbb{F}_{p^m}$, Freshman's dream allows us to do the following
$$(x + y)^p = x^p + y^p.$$
- If α is a primitive element in $\mathbb{F}_{p^m}$, then $\{1, \alpha, \ldots, \alpha^{m-1}\}$ builds a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$. That is, any $x \in \mathbb{F}_{p^m}$ can be written as
$$x = \sum_{i=0}^{m-1} x_i \alpha^i.$$
- For any basis $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$, we can define the expansion map
$$\exp_\Gamma : \mathbb{F}_{p^m} \to \mathbb{F}_p^m, \quad x = \sum_{i=1}^{m} x_i \gamma_i \mapsto \exp_\Gamma(x) = (x_1, \ldots, x_m).$$

Some more Details

This is a very compact form of the first chapter in `https://user.math.uzh.ch/weger/CT.pdf`. Thus, if you are interested in the proofs, have a look there!

Definition 7.1. A *finite field* is a field which is finite in size.

A first example is the prime field $\mathbb{F}_p$:

Theorem 7.2 (Prime Field)**.** *For every prime p the integer residue ring $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$ is a field, denoted by $\mathbb{F}_p$.*

Example 7.3. *The finite field of order 3 is given by $\mathbb{F}_3 = \mathbb{Z}/3\mathbb{Z} = \{0, 1, 2\}$. $\mathbb{Z}/4\mathbb{Z}$ is not a finite field, as 2 has no multiplicative inverse.*

Theorem 7.4. *Let $\mathbb{F}$ be a finite field with q elements. Then there exist a subfield $\mathbb{F}_p$ of $\mathbb{F}$ which is a prime field.*

Definition 7.5. Let $\mathbb{F}$ be a finite field with prime subfield $\mathbb{F}_p$. Then p is called the *characteristic* of $\mathbb{F}$.

Lemma 7.6. *Let $\mathbb{F}$ be a finite field, then there exist $p \in \mathcal{P}$ and $m \in \mathbb{N}$, such that $|\mathbb{F}| = p^m$.*

Let $\mathbb{F}$ be a finite field of order $q = p^m$, then m is called the *extension degree* of $\mathbb{F}$ over $\mathbb{F}_p$.
To construct the finite field $\mathbb{F}$ with $q = p^m$ many elements, we consider polynomials over $\mathbb{F}_p$:

$$\mathbb{F}_p[x] = \left\{ \sum_{i \geq 0} f_i x^i \mid f_i \in \mathbb{F}_p \right\}.$$

The addition and multiplication of two polynomials is performed in the usual way, taking modulo p for the coefficients. Then, $\mathbb{F}_p[x]$ forms a ring and is called the polynomial ring over $\mathbb{F}_p$.

Let $g(x) \in \mathbb{F}_p[x]$ be an irreducible polynomial of degree m and let us consider

$$\mathbb{F}_p[x]/\langle g(x) \rangle = \{ f(x) + h(x)g(x) \mid f(x), h(x) \in \mathbb{F}_p[x] \}.$$

That is, we quotient by the ideal generated by $g(x)$ (similar to integer residue rings, setting all multiples of $g(x)$ to zero and only considering the remainder $f(x)$.) This is well-defined, as a monic polynomial $q(x)$ of degree m can uniquely be written as $q(x) = g(x)h(x) + f(x)$, where $\deg(f) < m$. Thus, we may write $q(x) = f(x) \mod g(x)$.

Hence, we may identify an element $q(x) = f(x) + h(x)g(x) \in \mathbb{F}_p[x]/\langle g(x) \rangle$ with its remainder $f(x)$ and the polynomial ring modulo $g(x)$ consists of all polynomials of degree up to $m - 1$:

$$\mathbb{F}_p[x]/\langle g(x) \rangle = \left\{ f(x) = \sum_{i=0}^{m-1} f_i x^i \mid f_i \in \mathbb{F}_p \right\}.$$

The size of $\mathbb{F}_p[x]/\langle g(x) \rangle$ is thus p^m, as we consider all polynomials of degree up to $m - 1$.

Theorem 7.7. *Let $p \in \mathcal{P}, m$ a positive integer and $g(x) \in \mathbb{F}_p[x]$ an irreducible polynomial of degree m. Then, $\mathbb{F}_p[x]/\langle g(x)\rangle$ is a finite field with p^m elements.*

Example 7.8. *Let us consider $g(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$. Then,*

$$\mathbb{F}_2[x]/\langle g(x)\rangle = \{0, 1, x, x+1\}.$$

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Table 3: Addition in $\mathbb{F}_2[x]/\langle(x^2 + x + 1)\rangle$

·	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Table 4: Multiplication in $\mathbb{F}_2[x]/\langle(x^2 + x + 1)\rangle$

Let us consider $\mathbb{F}$ a finite field with q elements. By the definition of a field, we have that $\mathbb{F}^\star = \mathbb{F} \setminus \{0\}$ is an abelian multiplicative group. This turns out to be a cyclic group.

Definition 7.9. Let $\mathbb{F}$ be a finite field with q elements. An element $\alpha \in \mathbb{F}$ is called *primitive element*, if

$$\langle\alpha\rangle = \{\alpha^i \mid i \in \mathbb{N}\} = \mathbb{F}^\star.$$

Theorem 7.10. *Let $\mathbb{F}$ be a finite field, then $\mathbb{F}^\star$ is a cyclic multiplicative subgroup.*

Example 7.11. *In $\mathbb{F}_5$, we have 2 is a primitive element, as $\mathbb{F}_5^\star = \{2^0, 2^1, 2^3, 2^2\}$.*

Once we are given a primitive element, we can also construct all others and any element of order $d \mid (q-1)$.

Instead of considering elements in the finite field $\mathbb{F}_{p^m}$ as polynomials over $\mathbb{F}_p$ modulo an irreducible polynomial $f(x)$ of degree m, we may always use a root α of $f(x)$ to represent the elements of $\mathbb{F}_{p^m}$.

Definition 7.12. Let p be a prime and m a positive integer. Let $f(x) = \sum_{i=0}^{m-1} f_i x^i + x^m \in \mathbb{F}_p[x]$ be an irreducible polynomial of degree m and let α be a root of $f(x)$. Then, $\mathbb{F}_p$ *adjoin* α is

$$\mathbb{F}_p(\alpha) = \left\{ \sum_{i=0}^{m-1} a_i \alpha^i \mid a_i \in \mathbb{F}_p \right\}.$$

Similarly, we can consider all $a_0 \in \mathbb{F}_p$ and $a_i = 0$ for all $i > 0$ to show that $\mathbb{F}_p \subset \mathbb{F}_p(\alpha)$. Also the other properties can be easily checked.

Theorem 7.13. *Let p be a prime and m a positive integer. Let $f(x) = \sum_{i=0}^{m-1} f_i x^i + x^m \in \mathbb{F}_p[x]$ be an irreducible polynomial of degree m and let α be a root of $f(x)$. Then,*

$$\mathbb{F}_p[x]/\langle f(x) \rangle \cong \mathbb{F}_p(\alpha).$$

Instead of considering any irreducible polynomial, one can also consider a primitive polynomial.

Definition 7.14. Let $\alpha \in \mathbb{F}_{p^m}$ be a primitive element. The minimal polynomial of α is called *primitive polynomial.*

To check whether an irreducible polynomial $f(x)$ of degree m is primitive, we can equivalently check if the smallest positive integer n such that $f(x)$ divides $x^n - 1$ is $n = p^m - 1$.

Example 7.15. *Let us consider $\mathbb{F}_3$. The polynomial $x^2 + 1$ is irreducible, but not primitive as it divides $x^4 - 1$. A primitive polynomial of degree 2 would for example be $x^2 + 2x + 2$.*

If we have identified a primitive polynomial $f(x) \in \mathbb{F}_p[x]$ and a root α, we can immediately write all elements of $\mathbb{F}_{p^m}^\star$ as powers of α.

Example 7.16. *We are allowed to use any irreducible polynomial, as*

$$\mathbb{F}_9 \cong \mathbb{F}_3[x]/\langle x^2 + 1 \rangle \cong \mathbb{F}_3[x]/\langle x^2 + 2x + 2 \rangle.$$

Let us denote by α a root of $x^2 + 2x + 2$, i.e., $\alpha^2 = \alpha + 1$ and β a root of $x^2 + 1$, i.e., $\beta^2 = 2$. While we can write

$$\mathbb{F}_9 \cong \mathbb{F}_3(\alpha) \cong \mathbb{F}_3(\beta),$$

only α generates the multiplicative group $\mathbb{F}_9^\star$:

$$\alpha^0 = 1, \alpha^1 = \alpha, \alpha^2 = \alpha + 1, \alpha^3 = 2\alpha + 1, \alpha^4 = 2, \alpha^5 = 2\alpha, \alpha^6 = 2\alpha + 2, \alpha^7 = \alpha + 2.$$

We can also use a different representation, as $\mathbb{F}_{p^m} \cong \mathbb{F}_p^m$ as $\mathbb{F}_p$- vector space. For this we define the expansion map.

Definition 7.17. Let $\Gamma = \{\gamma_0, \ldots, \gamma_{m-1}\}$ be a basis of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$. Then the expansion map with respect to Γ is given by

$$\begin{aligned} \exp_\Gamma : \mathbb{F}_{p^m} &\to \mathbb{F}_p^m, \\ a = \sum_{i=0}^{m-1} a_i \gamma_i &\mapsto \exp_\Gamma(a) = (a_0, \ldots, a_{m-1}). \end{aligned}$$

This map is $\mathbb{F}_p$-linear, meaning that

- For $a, b \in \mathbb{F}_{p^m}$ we have $\exp_\Gamma(a+b) = \exp_\Gamma(a) + \exp_\Gamma(b)$,
- for $a \in \mathbb{F}_{p^m}$ and $\lambda \in \mathbb{F}_p$ we have that $\exp_\Gamma(\lambda a) = \lambda \exp_\Gamma(a)$.

As we also want to handle $\exp_\Gamma(ab)$ we need to introduce the multiplication matrix.

For this we will focus on a basis $\Gamma = \{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ of $\mathbb{F}_p^m$ over $\mathbb{F}_p$, we call such a basis a *polynomial basis.* Note that $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ is a basis of $\mathbb{F}_p^m$ over $\mathbb{F}_p$, if and only if α is a root of $f(x) \in \mathbb{F}_p[x]$, which is irreducible and of degree m.

Let α be the root of an irreducible polynomial $f(x) = \sum_{i=0}^{m-1} f_i x^i + x^m \in \mathbb{F}_p[x]$ of degree m and consider

$$\begin{aligned}
&\mathbb{F}_p[x]/\langle f(x)\rangle &&\overset{\text{ev}_\alpha}{\cong} \mathbb{F}_p(\alpha) &&\overset{\exp_\Gamma}{\cong} \mathbb{F}_p^m,\\
&a(x) = \sum_{i=0}^{m-1} a_i x^i &&\mapsto a = \sum_{i=0}^{m-1} a_i \alpha^i &&\mapsto \exp_\Gamma(a) = (a_0, \ldots, a_{m-1}),\\
&a(x) + b(x) &&\mapsto a + b &&\mapsto \exp_\Gamma(a) + \exp_\Gamma(b).
\end{aligned}$$

However, what happens to the multiplication? We can compute $a(x)b(x) \mod f(x)$ and $ab \in \mathbb{F}_p(\alpha)$, thus we need to figure out what $\exp_\Gamma(ab)$ is in terms of $\exp_\Gamma(a)$ and $\exp_\Gamma(b)$.

For this we define the multiplication matrix for $b \in \mathbb{F}_p(\alpha)$ via the basis Γ as

$$M_\Gamma(b) = \begin{pmatrix} \exp_\Gamma(b) \\ \exp_\Gamma(\alpha b) \\ \vdots \\ \exp_\Gamma(\alpha^{m-1} b) \end{pmatrix}$$

and define

$$\exp_\Gamma(a) \circ \exp_\Gamma(b) = \exp_\Gamma(a) M_\Gamma(b).$$

Note that for $b = \alpha$, the multiplication matrix $M_\Gamma(\alpha)$ is the companion matrix of $f(x)$ as

$$M_\Gamma(\alpha) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ -f_0 & -f_1 & -f_2 & \cdots & -f_{m-1} \end{pmatrix}.$$

Example 7.18. *Let us consider $\mathbb{F}_8 \cong \mathbb{F}_2(\alpha)$ where α is a primitive root and satisfies $\alpha^3 = \alpha + 1$ as the primitive polynomial over $\mathbb{F}_2$ is given by $x^3 + x + 1$. Thus, we have the polynomial basis $\Gamma = \{1, \alpha, \alpha^2\}$ of $\mathbb{F}_8$ over $\mathbb{F}_2$. Let $a = \alpha^2 + 1$ and $b = \alpha + 1$. We can easily expand them to $\mathbb{F}_2^3$ as*

$$exp_\Gamma(a) = (1, 0, 1), \quad exp_\Gamma(b) = (1, 1, 0).$$

We want to multiply a with b and over $\mathbb{F}_8$ we can easily check that $ab = \alpha^2$ which is expanded to

$$exp_\Gamma(ab) = (0, 0, 1).$$

We compute

$$M_\Gamma(b) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

and check

$$M_\Gamma(b) exp_\Gamma(a) = (0, 0, 1) = exp_\Gamma(ab).$$

To summarize, we may view the finite field $\mathbb{F}_{p^m}$ in three different ways:

Let p be a prime and m a positive integer. Let $f(x) = \sum_{i=0}^{m-1} f_i x^i + x^m \in \mathbb{F}_p[x]$ be an irreducible polynomial of degree m and let α be a root of $f(x)$.

$\mathbb{F}_p[x]/\langle f(x)\rangle$	$\mathbb{F}_p(\alpha)$	$\mathbb{F}_p^m$
$a(x) = \sum_{i=0}^{m-1} a_i x^i$	$a = \sum_{i=0}^{m-1} a_i \alpha^i$	$(a_0, \ldots, a_{m-1})$
$a(x) + b(x) \mod f(x)$	$a + b$	$\exp_\Gamma(a) + \exp_\Gamma(b)$
$a(x) \cdot b(x) \mod f(x)$	$a \cdot b$	$\exp_\Gamma(a) \circ \exp_\Gamma(b)$

Example 7.19. *Let us consider again*

$$\mathbb{F}_9 \cong \mathbb{F}_3[x]/\langle x^2 + 1\rangle \cong \mathbb{F}_3[x]/\langle x^2 + 2x + 2\rangle,$$

and denote by α a root of $x^2 + 2x + 2$, i.e., $\alpha^2 = \alpha + 1$ and β a root of $x^2 + 1$, i.e., $\beta^2 = 2$. Recall that for α a primitive element over $\mathbb{F}_q$ and β of order z, we have that $\beta = \alpha^{i(q-1)/z}$ for some $i \in \{1, \ldots, z\}$ and $gcd(i, (q-1)/z) = 1$. Thus, assuming $i = 1$, we get that $\beta = \alpha^2 = \alpha + 1$, and can convert the different representations:

Definition 7.20. The map $\varphi : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}, \alpha \mapsto \alpha^q$ is called *Frobenius map.*

Note that for any $x \in \mathbb{F}_{q^m}$ we have $x \in \mathbb{F}_q$ if and only if $x^q = x$.

$\mathbb{F}_3(\beta)$	0	1	2	β	2β	$\beta+1$	$\beta+2$	$2\beta+1$	$2\beta+2$
$\mathbb{F}_3(\alpha)$	0	1	2	$\alpha+1$	$2\alpha+2$	$\alpha+2$	α	2α	$2\alpha+1$
$\langle\alpha\rangle$		α^0	α^4	α^2	α^6	α^7	α^1	α^5	α^3

Definition 7.21. Let $\mathbb{F}_{q^m}$ and $\mathbb{F}_q$ be the finite field with q^m, respectively q, elements. The *trace* map is defined as

$$\begin{aligned}\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q} : \mathbb{F}_{q^m} &\to \mathbb{F}_q,\\ \alpha &\mapsto \sum_{i=0}^{m-1} \alpha^{q^i}.\end{aligned}$$

It might not directly be clear why this map sends elements from $\mathbb{F}_{q^m}$ to $\mathbb{F}_q$. We have a simple test to check whether an element of $x \in \mathbb{F}_{q^m}$ is actually living in the subfield $\mathbb{F}_q$: check if $x^q = x$.

Thus, we compute

$$\left(\sum_{i=0}^{m-1} \alpha^{q^i}\right)^q = \sum_{i=0}^{m-1} \alpha^{q^i \cdot q} = \sum_{i=0}^{m-1} \alpha^{q^{i+1}} = \sum_{i=0}^{m-1} \alpha^{q^i},$$

where we have used that $\alpha^{q^m} = \alpha$.

The traces possess many interesting properties, which we leave as an exercise:

Theorem 7.22. *Let $\alpha, \beta \in \mathbb{F}_{q^m}$ and $\lambda \in \mathbb{F}_q$. Then*

1. $Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha+\beta) = Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha) + Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\beta)$,
2. $Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\lambda\alpha) = \lambda Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha)$,
3. $Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\lambda) = m\lambda$,
4. $Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha^q) = Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha)$.

We may ask if $\mathbb{F}_{p^m}$ has also other subfields than $\mathbb{F}_p$.

Theorem 7.23. *Let $\mathbb{F}_q$ be a field with $q = p^m$ elements. $\mathbb{F}_q$ has a subfield of order p^r, if and only if $r \mid m$.*

And we conclude this short recap on finite fields with Freshman's dream.

Theorem 7.24 (Freshman's dream)**.** *For any $a, b \in \mathbb{F}_{p^m}$ we have that*

$$(a+b)^p = a^p + b^p.$$

7.2 Codes

In this section, we give the basics of algebraic coding theory. This is a short version of Chapter 2 in `https://user.math.uzh.ch/weger/CT.pdf`.

Let us fix that $\mathbb{F}_q$ will denote the finite field of q elements, where q is a prime power.

Definition 7.25 (Linear Code)**.** Let $1 \leq k \leq n$ be integers. Then, an $[n,k]_q$ *linear code* $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$.

As $\mathcal{C}$ is linear, it must have some basis, which allows us to represent it compactly. In fact, linear codes allow for an easy representation through their *generator matrices*, which have the code as an image.

Definition 7.26 (Generator Matrix)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]_q$ linear code. Then, a matrix $G \in \mathbb{F}_q^{k\times n}$ is called a *generator matrix* of $\mathcal{C}$ if

$$\mathcal{C} = \left\{ xG \mid x \in \mathbb{F}_q^k \right\},$$

that is, the rows of G form a basis of $\mathcal{C}$.

We will often write $\langle G \rangle$ to denote the code generated by the rows G.

One can also represent a code through a matrix H, which has the code as kernel.

Definition 7.27 (Parity-Check Matrix)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]_q$ linear code. Then, a matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ is called a *parity-check matrix* of $\mathcal{C}$, if

$$\mathcal{C} = \left\{ y \in \mathbb{F}_q^n \mid yH^\top = 0 \right\}.$$

For any $x \in \mathbb{F}_q^n$, we call $xH^\top$ the *syndrome* of x through H.

Let $H \in \mathbb{F}_q^{(n-k)\times n}$ be a parity-check matrix and assume $x \in \mathbb{F}_q^n$ is unknown. Then, we get a system of $n-k$ linear equations in x_i from $Hx^\top = s^\top$:

$$\begin{aligned} \sum_{i=1}^{n} h_{1,i}x_i &= s_1 \\ &\vdots \\ \sum_{i=1}^{n} h_{n-k,i}x_i &= s_{n-k}. \end{aligned}$$

These equations are called *parity-check equations* or *syndrome equations.*

Consider our original problem, where we have received $r = c + e \in \mathbb{F}_q^n$, with $c \in \mathcal{C}$ the sent codeword and e an error vector added by the channel. By computing the syndrome of r via the parity-check matrix H of $\mathcal{C}$, we get

$$s = rH^\top = (c+e)H^\top = cH^\top + eH^\top = eH^\top,$$

i.e., we see that the received word is erroneous, and get an equation only depending on the error vector.

Since $\mathcal{C} = \mathrm{Im}(G) = \ker(H^\top)$, we also get a relation between the two matrices, namely

$$GH^\top = 0.$$

For $x, y \in \mathbb{F}_q^n$ let us denote by $\langle x, y \rangle$ the standard inner product, i.e.,

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i.$$

Then, we can define the dual of an $[n,k]_q$ linear code $\mathcal{C}$ as the orthogonal space of $\mathcal{C}$.

Definition 7.28 (Dual Code)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]_q$ linear code. The *dual code* $\mathcal{C}^\perp$ is an $[n, n-k]_q$ linear code, defined as

$$\mathcal{C}^\perp = \{x \in \mathbb{F}_q^n \mid \langle x, y \rangle = 0 \; \forall \, y \in \mathcal{C}\}.$$

Proposition 7.29. *Let q be a prime power and $k \leq n$ be positive integers. Let $\mathcal{C}$ be an $[n,k]_q$ linear code. Then $(\mathcal{C}^\perp)^\perp = \mathcal{C}$.*

For $x \in \mathbb{F}_q^n$ and $S \subseteq \{1, \ldots, n\}$ we denote by x_S the vector consisting of the entries of x indexed by S. While for $A \in \mathbb{F}_q^{k \times n}$, we denote by A_S the matrix consisting of the columns of A indexed by S. Similarly, we denote by $\mathcal{C}_S$ the code consisting of the codewords c_S, i.e.,

$$\mathcal{C}_S = \{c_s \mid c \in \mathcal{C}\}.$$

An $[n,k]_q$ linear code can be completely defined by having access only to (correctly chosen) k positions. The following concept characterizes such defining sets.

Definition 7.30 (Information Set)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]_q$ linear code. Then, a set $I \subset \{1, \ldots, n\}$ of size k is called an *information set* of $\mathcal{C}$ if

$$| \, \mathcal{C} \, | = | \, \mathcal{C}_I \, | \, .$$

Exercise 7.31. *How many information sets can an $[n,k]_q$ linear code have at most?*

Proposition 7.32. *Let $\mathcal{C}$ be an $[n,k]_q$ linear code, I an information set and let G be a generator matrix . The matrix G_I is an invertible matrix.*

Proposition 7.33. *Let $\mathcal{C}$ be an $[n,k]_q$ linear code, I an information set and H a parity-check matrix. If $I^C := \{1, \ldots, n\} \setminus I$ is the complement set of I, then, H_{I^C} is an invertible matrix.*

Exercise 7.34. *Let $\mathcal{C}$ be the code generated by $G \in \mathbb{F}_5^{2 \times 4}$, given as*

$$G = \begin{pmatrix} 1 & 3 & 2 & 3 \\ 0 & 4 & 4 & 3 \end{pmatrix}.$$

Determine all information sets of this code.

Since G_I and H_{I^C} are invertible, we can apply row operations U, respectively U' to get

$$(UG)_I = \mathrm{Id}_k, \quad (U'H)_{I^C} = \mathrm{Id}_{n-k},$$

which leads to the following definition of a *systematic form.*

Definition 7.35 (Systematic Form)**.** Let $k \leq n$ be positive integers and $\mathcal{C}$ be an $[n,k]_q$ linear code. Then, there exist some $n \times n$ permutation matrix P and some invertible matrix $U \in \mathbb{F}_q^{k\times k}$ that bring G in *systematic form*, i.e.,

$$UGP = \begin{pmatrix} \mathrm{Id}_k & A \end{pmatrix},$$

where $A \in \mathbb{F}_q^{k\times(n-k)}$. Similarly, there exist some $n \times n$ permutation matrix P' and some invertible matrix $U' \in \mathbb{F}_q^{(n-k)\times(n-k)}$, that bring H into systematic form, i.e.,

$$U'HP' = \begin{pmatrix} B & \mathrm{Id}_{n-k} \end{pmatrix},$$

where $B \in \mathbb{F}_q^{(n-k)\times k}$.

As we have not introduced permutation equivalence yet, we may think of the standard form as follows: Let $I \subseteq \{1,\dots,n\}$ be an information set and G a generator matrix of $\mathcal{C}$, then there exists $U \in \mathrm{GL}_q(k)$ such that

$$(UG)_I = \mathrm{Id}_k, \quad \text{and} \quad (UG)_{I^C} = A,$$

for some $A \in \mathbb{F}_q^{k\times(n-k)}$. We will later see that permuting the identity matrix to be at the first k coordinates, will not change the underlying structure.

Given a generator matrix, one can easily find a parity-check matrix of the code.

Proposition 7.36. *Let $\mathcal{C}$ be an $[n,k,]_q$ linear code and G be a generator matrix. If $G = \begin{pmatrix} Id_k & A \end{pmatrix}$, for some $A \in \mathbb{F}_q^{k\times(n-k)}$, then $H = \begin{pmatrix} -A^\top & Id_{n-k} \end{pmatrix}$ is a parity-check matrix of $\mathcal{C}$.*

If the identity matrix is not in the first k coordinates, we think of Proposition 7.36 as $G_I = \mathrm{Id}_k$ and $G_{I^C} = A$ and thus $H_{I^C} = \mathrm{Id}_{n-k}$ and $H_I = -A^T$.

As we are interested in the amount of positions which are erroneous, the most natural weight to consider is the *Hamming metric*.

Definition 7.37 (Hamming Metric)**.** Let n be a positive integer. For $x \in \mathbb{F}_q^n$, the *Hamming weight* of x is given by the number of non-zero positions, i.e.,

$$\mathrm{wt}_H(x) = \mid \{i \in \{1,\dots,n\} \mid x_i \neq 0\} \mid .$$

For $x, y \in \mathbb{F}_q^n$, the *Hamming distance* between x and y is given by the number of positions in which they differ, i.e.,

$$d_H(x,y) = \mid \{i \in \{1,\dots,n\} \mid x_i \neq y_i\} \mid .$$

Note that the Hamming distance is induced from the Hamming weight, that is $d_H(x,y) = \mathrm{wt}_H(x-y)$. We can also consider the minimum distance of a code, i.e., the smallest distance between any two distinct codewords.

Definition 7.38 (Minimum Distance)**.** Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$. The *minimum Hamming distance* of $\mathcal{C}$ is denoted by $d_H(\mathcal{C})$ and given by

$$d_H(\mathcal{C}) = \min\{d_H(x,y) \mid x,y \in \mathcal{C},\ x \neq y\} = \min\{\mathrm{wt}_H(c) \mid c \in \mathcal{C}, c \neq 0\}.$$

The minimum Hamming distance of a code turns out to be a very important parameter. Thus, whenever the minimum Hamming distance $d = d_H(\mathcal{C})$ is known, we say $\mathcal{C}$ is an $[n,k,d]_q$ linear code.

When defining balls in a certain metric, we have to provide the radius and the center, e.g. we may define the Hamming ball of radius r and center x as

$$B_H(r,n,q,x) = \{y \in \mathbb{F}_q^n \mid d_H(x,y) \leq r\}.$$

However, to determine the size of such balls, we observe that the Hamming metric is translation invariant.

Proposition 7.39. *Let q be a prime power and $r \leq n$ be positive integers. For any $x, x' \in \mathbb{F}_q^n$ we have*

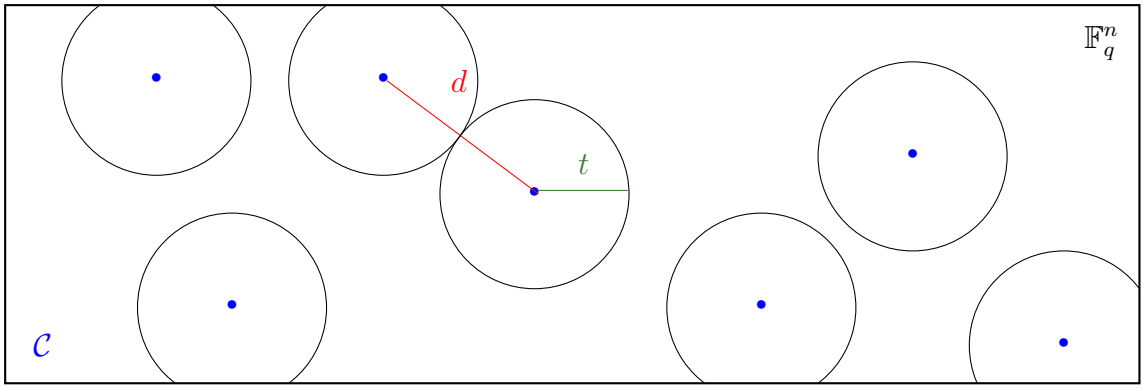
$$|B_H(r,n,q,x)| = |B_H(r,n,q,x')| = |\{y \in \mathbb{F}_q^n \mid wt_H(y) \leq r\}| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

We denote by $d_H(x,\mathcal{C})$ the minimal distance between $x \in \mathbb{F}_q^n$ and a codeword in $\mathcal{C}$.

We say that a code can *correct* up to t errors, if for all $r \in \mathbb{F}_q^n$ with $d_H(r,\mathcal{C}) \leq t$, there exists at most one $c \in \mathcal{C}$, such that $d_H(r,c) \leq t$. The parameter t is then called the *error correction capability* of the code. Equivalently, $\mathcal{C}$ can correct t errors, if the balls of radius t around any distinct codewords $c \neq c'$ are disjoint: $B_H(t,n,q,c) \cap B_H(t,n,q,c') = \emptyset$.

If we are given the code $\mathcal{C} \subseteq \mathbb{F}_q^n$ and depict its codewords as points, then the shortest distance between two of them is given by $d_H(\mathcal{C}) = d$. To find the error correction capability, we want to draw balls around the codewords, with radius as large as possible, but such that the balls do not intersect. This results in the radius

$$t = \left\lceil \frac{d-1}{2} \right\rceil.$$



Theorem 7.40. *Let $\mathcal{C}$ be an $[n,k,d]_q$ linear code. Then $t = \lceil \frac{d-1}{2} \rceil$ is the error correction capability of the code.*

The following theorem let's us compute the minimum distance from the parity-check matrix H.

Theorem 7.41. *Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]_q$ linear code. Let H be a parity-check matrix of $\mathcal{C}$. Then, $\mathcal{C}$ has minimum distance d if and only if every $d-1$ columns of H are linearly independent and there exist d columns, which are linearly dependent.*

Let $\mathcal{C}$ be an $[n,k,d]_q$ linear code with $\langle G \rangle = \mathcal{C} = \ker(H^\top)$.

- The parameter n is called the *length* of the code.
- The parameter k is called the *dimension* of the code.
- The elements in the code are called *codewords*.
- The parameter $r = n - k$ is called the *redundancy*.
- The parameter $R = k/n$ is called the *rate* of the code.
- The matrix G is called a *generator matrix* of the code.
- The matrix H is called a *parity-check matrix* of the code.
- The vector $s = xH^\top$ is called the *syndrome* of x.
- The code $\mathcal{C}^\perp$ is called the *dual code* of $\mathcal{C}$.
- A set I with $G_I \in \mathrm{GL}_q(k)$ is called *information set*.
- The parameter d is called the *minimum Hamming distance* of the code.
- The parameter $t = \lceil \frac{d-1}{2} \rceil$ is called the *error correction capability* of the code.

7.3 Complexity Classes

Let $\mathcal{P}$ denote a problem. In order to estimate how hard it is to solve $\mathcal{P}$ we have two main complexity classes.

Definition 7.42. P denotes the class of problems that can be solved by a deterministic Turing machine in polynomial time.

The concept of deterministic and non-deterministic Turing machines will exceed the scope of this chapter, just note that "can be solved by a deterministic Turing machine in polynomial time" is the same as our usual "can be solved in polynomial time".

Example 7.43. *Given a list S of n integers and an integer k, determine whether there is an integer $s \in S$ such that $s > k$? Clearly, this can be answered by going through the list and checking for each element whether it is greater than k, thus it has running time at most n and this problem is in P.*

Definition 7.44. NP denotes the class of problems that can be solved by a non-deterministic Turing machine in polynomial time.

Thus, in contrary to the popular belief that NP stands for non-polynomial time, it actually stands for non-deterministic polynomial time. The difference is important: all problems in P live inside NP!

To understand NP better, we might use the equivalent definition: A problem $\mathcal{P}$ is in NP if and only if one can check that a candidate is a solution to $\mathcal{P}$ in polynomial time.

The example from before is thus also clearly in NP, since if given a candidate a, we can check in polynomial time whether $a \in S$ and whether $a > k$.

There are, however, interesting problems which are in NP, but we do not know whether they are in P. Let us change the previous example a bit.

Example 7.45. *Given a list S of n integers and an integer k, is there a set of integers $T \subseteq S$, such that $\sum_{t \in T} t = k$? Since there are exponentially many subsets of S, there is no known algorithm to solve this problem in polynomial time and thus, we do not know whether it lives in P. But, if given a candidate T, we can check in polynomial time if all $t \in T$ are also in S and if $\sum_{t \in T} t = k$, which clearly places this problem inside NP.*

The most important complexity class, for us, will be that of NP-hard problems. In order to define this class, we first have to define polynomial-time reductions.

A polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$ follows the following steps:

1. take any instance I of $\mathcal{R}$,
2. transform I to an instance I' of $\mathcal{P}$ in polynomial time,

3. assume that (using an oracle) you can solve $\mathcal{P}$ in the instance I' in polynomial time, getting the solution s',

4. transform the solution s' in polynomial time to get a solution s of the problem $\mathcal{R}$ in the input I.

The existence of a polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$, informally speaking, means that if we can solve $\mathcal{P}$, we can also solve $\mathcal{R}$ and thus solving $\mathcal{P}$ is at least as hard as solving $\mathcal{R}$.

Definition 7.46. $\mathcal{P}$ is NP-hard if for every problem $\mathcal{R}$ in NP, there exists a polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$.

Informally speaking, this class contains all problems which are at least as hard as the hardest problems in NP.

Example 7.47. *One of the most famous examples for an NP-hard problem is the subset sum problem: given a set of integers S, is there a non-empty subset $T \subseteq S$, such that $\sum_{t \in T} t = 0$?*

We want to remark here, that NP-hardness is only defined for *decisional* problems, that are problems of the form "decide whether there exists.." and not for *computational/search* problems, that are problems of the form "find a solution..". However, considering for example the SDP, in its decisional version, it asks whether there exists error vector e with certain conditions. If one could solve the computational problem, that is to actually find such an error vector e in polynomial time, then one would also be able to answer the decisional problem in polynomial time. Thus, not being very rigorous, we call also the computational SDP NP-hard.

In order to prove that a problem $\mathcal{P}$ is NP-hard, fortunately we do not have to give a polynomial-time reduction to *every* problem in NP: there are already problems which are known to be NP-hard, thus it is enough to give a polynomial-time reduction from an NP-hard problem to $\mathcal{P}$.

Finally, NP-completeness denotes the intersection of NP-hardness and NP.

Definition 7.48. A problem $\mathcal{P}$ is NP-complete, if it is NP-hard and in NP.

7.4 Cost of Algorithms

To estimate the cost of algorithms, we use the big-O notation.

Definition 7.49. Let $f(n), g(n)$ be functions over the reals. We write $f(n) \in \mathcal{O}(g(n))$ to denote that there exists a positive real constant N such that $|f(n)| < Ng(n)$ for all $n > n_0$, meaning that if n grows, $f(n)$ will not grow faster than $g(n)$.

Example 7.50. *For example* $n + 2n^2 \in \mathcal{O}(n^2)$, *while* $2^{n/2}n^5 + n^2 2^n \in \mathcal{O}\left(2^n\right)$.

Thus, we should read it as "we ignore all lower term".

We say that $f(n)$ is polynomial if $f(n) \in \mathcal{O}\left(n^c\right)$ for some constant c.

In this lecture we are dealing with objects over $\mathbb{F}_q$, where we can add, multiply or invert elements with different costs:

- adding in $\mathbb{F}_q$ has a cost in $\mathcal{O}\left(\lceil \log_2(q) \rceil\right)$,
- multiplication in $\mathbb{F}_q$ has a cost in $\mathcal{O}\left(\lceil \log_2(q) \rceil^2\right)$,
- inverting an element in $\mathbb{F}_q$ has a cost in $\mathcal{O}\left(\lceil \log_2(q) \rceil^3\right)$.

In our algorithms we are also interested in doing matrix multiplication. Here school-book multiplication will be enough. That is for $A \in \mathbb{F}_q^{k \times n}, B \in \mathbb{F}_q^{n \times m}$ computing AB has a cost in $\mathcal{O}(mnk)$.

If we have to construct a list $\mathcal{L}$ containing L elements and each element costs a operations, we get that the construction of the list has a cost in $\mathcal{O}(La)$.

If for example $L \in \mathcal{O}\left(2^n\right)$ and $a \in \mathcal{O}(n)$, then clearly we get a cost in $\mathcal{O}(L)$.

To go through lists $\mathcal{L}$ costs $\mathcal{O}(L)$. We could also sort it (or use hashing), costing $\mathcal{O}(L \log_2(L))$, but then going through the list becomes less costly with $\mathcal{O}(\log_2(L))$.

If we have two lists $\mathcal{L}_1, \mathcal{L}_2$ and we want to go find a collision between them, e.g. $(a, x) \in \mathcal{L}_1$ and $(a, y) \in \mathcal{L}_2$, we multiply the cost of going through $\mathcal{L}_1 \times \mathcal{L}_2$ by the probability p of having a collision: $\mathcal{O}(L^2 p)$.

While the construction of a new list $\mathcal{L} = \mathcal{L}_1 \times \mathcal{L}_2$ can be made faster using sorting, if we later have to operate on all elements in $\mathcal{L}$, the cost is still given by the expected size of $\mathcal{L}$.

7.5 Cryptography

As coding theory is the art of *reliable* communication, this goes hand in hand with cryptography, the art of *secure* communication. In cryptography we differ between two main branches, symmetric cryptography and asymmetric cryptography.

In *symmetric* cryptography there are the two parties that want to communicate with each other and prior to communication have exchanged some key, that will enable them a secure communication. Such secret key exchange might be performed using protocols such as the Diffie-Hellman key exchange [10], which itself lies in the realm of asymmetric cryptography.

More mathematically involved is the branch of *asymmetric* cryptography, where the two parties do not share the same key. In this survey we will focus on two main subjects of asymmetric cryptography, that were also promoted by the NIST standardization call [9], namely public-key encryption (PKE) schemes and digital signature schemes.

Many of these cryptographic schemes seem very abstract when discussed in generality. To get a grasp of the many definitions and concepts, we will also provide some easy examples. First of all, let us recall the definition of a hash function. A *hash function* is a function that compresses the input value to a fixed length. In addition, we want that it is computationally hard to reverse a hash function and also to find a different input giving the same hash value. In this chapter, we denote a publicly known hash function by Hash.

Let us have a look at *public-key encryption (PKE)* schemes. A PKE consists of three steps:

1. key generation,
2. encryption,
3. decryption.

The main idea is that one party, usually called Alice, constructs a *secret key* $\mathcal{S}$ and a connected *public key* $\mathcal{P}$. The public key, as the name suggests, is made publicly known, while the secret key is kept private.

This allows another party, usually called Bob, to use the public key to encrypt a *message* m by applying the public key, gaining the so called *cipher* c.

The cipher is now sent through the insecure channel to Alice, who can use her secret key $\mathcal{S}$ to decrypt the cipher and recover the message m.

An adversary, usually called Eve, can only see the cipher c and the public key $\mathcal{P}$. In order for a public-key encryption scheme to be considered secure, it should be infeasible for Eve to recover from c and $\mathcal{P}$ the message m. This also implies that the public key should not reveal the secret key.

What exactly does infeasible mean, however? This is the topic of *security*. For a cryptographic scheme, we define its *security level* to be the average number of binary operations needed for an adversary to break the cryptosystem, that means either to recover the message (called *message recovery*) or the secret key (called *key recovery*).

Table 5: Public-Key Encryption

ALICE		BOB
KEY GENERATION		
Construct a secret key $\mathcal{S}$		
Construct a connected public key $\mathcal{P}$		
	$\xrightarrow{\mathcal{P}}$	
		ENCRYPTION
		Choose a message m
		Encrypt the message $c = \mathcal{P}(m)$
	$\xleftarrow{c}$	
DECRYPTION		
Decrypt the cipher $m = \mathcal{S}(c)$		

Usual security levels are $2^{80}, 2^{128}, 2^{256}$ or even 2^{512}, meaning for example that an adversary is expected to need at least 2^{80} binary operations in order to reveal the message. These are referred to as 80 bit, 128 bit, 256 bit, or 512 bit security levels.

Apart from the security of a PKE, one is also interested in the performance, including how fast the PKE can be executed and how much storage the keys require. Important parameters of a public-key encryption are

- the public key size,
- the secret key size,
- the ciphertext size,
- the decryption time.

These values are considered to be the *performance* of the public-key encryption. With 'size' we intend the bits that have to be sent or stored for this key, respectively for the cipher. Clearly, one prefers small sizes and a fast decryption.

References

[1] C. Aguilar Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. Bos, J.-C. Deneuville, A. Dion, P. Gaborit, J. Lacan, E. Persichetti, J.-M. Robert, P. Véron, and G. Zémor. Hamming Quasi-Cyclic (HQC). *NIST PQC Call for Proposals*, 2022. Round 4 Submission.

[2] C. Aguilar Melchor, T. Feneuil, N. Gama, S. Gueron, J. Howe, D. Joseph, A. Joux, E. Persichetti, T. H. Randrianarisoa, M. Rivain, and D. Yue. SDitH. In *First Round Submission to the additional NIST Postquantum Cryptography Call*, 2023.

[3] M. Baldi, A. Barenghi, S. Bitzer, P. Karl, F. Manganiello, A. Pavoni, G. Pelosi, P. Santini, J. Schupp, F. Slaughter, A. Wachter-Zeh, and V. Weger. CROSS. In *Second Round Candidate of the additional NIST Postquantum Cryptography Call*, 2023.

[4] G. Banegas, K. Carrier, A. Chailloux, A. Couvreur, T. Debris-Alazard, P. Gaborit, P. Karpman, J. Loyer, R. Niederhagen, N. Sendrier, B. Smith, and J.-P. Tillich. WAVE. In *First Round Submission to the additional NIST Postquantum Cryptography Call*, 2023.

[5] S. Barg. Some new NP-complete coding problems. *Problemy Peredachi Informatsii*, 30(3):23–28, 1994.

[6] A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 520–536. Springer, 2012.

[7] E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[8] L. Both and A. May. Decoding linear codes with high error rate and its impact for LPN security. In *International Conference on Post-Quantum Cryptography*, pages 25–46. Springer, 2018.

[9] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[10] W. Diffie and M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[11] L. Ducas, A. Esser, S. Etinski, and E. Kirshanova. Asymptotics and improvements of sieving for codes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 151–180. Springer, 2024.

[12] I. I. Dumer. Two decoding algorithms for linear codes. *Problemy Peredachi Informatsii*, 25(1):24–32, 1989.

[13] L. Engelberts, S. Etinski, and J. Loyer. Quantum sieving for code-based cryptanalysis and its limitations for isd. *Designs, Codes and Cryptography*, pages 1–34, 2025.

[14] A. Esser. Revisiting nearest-neighbor-based information set decoding. In *IMA International Conference on Cryptography and Coding*, pages 34–54. Springer, 2023.

[15] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 88–105. Springer, 2009.

[16] Q. Guo, T. Johansson, and V. Nguyen. A new sieving-style information-set decoding algorithm. *IEEE Transactions on Information Theory*, 2024.

[17] G. Kachigar and J.-P. Tillich. Quantum information set decoding algorithms. In *International Workshop on Post-Quantum Cryptography*, pages 69–89. Springer, 2017.

[18] E. Kirshanova. Improved quantum information set decoding. In *International Conference on Post-Quantum Cryptography*, pages 507–527. Springer, 2018.

[19] P. J. Lee and E. F. Brickell. An observation on the security of McElieces public-key cryptosystem. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 275–280. Springer, 1988.

[20] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

[21] A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{\jmath}(2^{0.054n})$. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 107–124. Springer, 2011.

[22] A. May and I. Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 203–228. Springer, 2015.

[23] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, Jan. 1978.

[24] A. Meurer. *A coding-theoretic approach to cryptanalysis*. PhD thesis, Ruhr-Universität Bochum, 2012.

[25] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[26] J. Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

[27] V. Weger, N. Gassner, and J. Rosenthal. A survey on code-based cryptography. *arXiv preprint arXiv:2201.07119*, 2022.