Problems Session 2:

Cryptanalysis Challenge

Recall that code-based cryptosystems mostly rely on the hardness of the decoding problem:

Data: A parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and a syndrome $\mathbf{s}^\top \stackrel{\text{def}}{=} \mathbf{H}\mathbf{x}^\top \in \mathbb{F}_q^{(n-k)}$ with $|\mathbf{x}| = t$.

Goal: Recover a solution $\mathbf{e}$ of Hamming weight t.

In order to better understand the security of cryptosystems based on this problem, it is important to assess its *practical* hardness with respect to today's computational resources. This is especially important for selecting concrete sets of parameters for the cryptosystems. In order to do that, cryptographers often generate a lot of challenges for different parameters, and ask other people to try and break them.

The goal of this practice session is to solve some challenges presented in `https://decodingchallenge.org/syndrome`. We will use the SageMath software, and most of the challenges will assume that we work over the binary field $\mathbb{F}_2$.

Setting up the Challenges

We provide some helper functions which helps you setting up the challenges. They are available at `https://maximebombar.fr/teaching_content/summer_schools/2024/zurich/helper_challenges.sage`.

Once you solved a challenge, please tell us!

Usage Example:

```
sage: load("helper_challenges.sage")
sage: download_challenges()
sage: H, s, w = parse_challenge("Challenges/SD_010")
sage: H
[1 0 0 0 0|1 1 0 0 1]
[0 1 0 0 0|1 1 1 1 0]
[0 0 1 0 0|0 1 0 0 1]
[0 0 0 1 0|1 1 0 0 1]
[0 0 0 0 1|1 0 1 1 1]
sage: s
(0, 1, 1, 1, 0)
sage: w
4
```

1 The Power of Linear Algebra: Prange Algorithm

Notation. For a matrix $\mathbf{A}$ with n columns, or a vector $\mathbf{x}$ of length n, and for $I \subset \{1, \dots, n\}$, we denote by $\mathbf{A}_I$ (resp. $\mathbf{x}_I$) the submatrix (resp. the subvector) formed by only keeping the columns of $\mathbf{A}$ (resp. the entries of $\mathbf{x}$) which are indexed by I.

Recall one of the most basic generic decoding algorithms, namely Prange algorithm

Algorithm 1: Prange algorithm

Input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k}$ and w that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$ for some $\mathbf{e}$ of weight w.
Output: $\mathbf{x} \in \mathbb{F}_2^n$ with $|\mathbf{x}| = w$ and $\mathbf{x}\mathbf{H}^\top = \mathbf{s}$.

1 Pick a set $I \subset \{1, \dots, n\}$ of size k until $\mathbf{H}_{I'}$ is full rank, where I' is the complement set of I.
2 Solve the linear system $\begin{cases} \mathbf{x}\mathbf{H}^\top = \mathbf{s} \\ \mathbf{x}_I = 0 \end{cases}$.
3 **if** $|\mathbf{x}| = w$ **then**
4   **return x**
5 **else**
6   Go back to Step 1

(Q1) Implement Prange algorithm using SageMath, and try to run it to solve the first challenges.

Hint: For solving the linear system of Step 2, you can invert the submatrix $\mathbf{H}_{I'}$.

2 It's Birthday Time: Dumer Algorithm

Recall from the lecture that Dumer algorithm is an improvement on the exhaustive search for decoding at the Gilbert-Varshamov distance (which is the case for those challenges), and for codes of rate close to 1. This algorithm will be used as a subroutine in section 3. Recall from the lecture that the goal is to split $\{1, \ldots, n\}$ into two random sets I_1 and I_2 of size $n/2$ so that

$$\mathbf{s} = \mathbf{e}\mathbf{H}^\top = \mathbf{e}_1\mathbf{H}_1^\top + \mathbf{e}_2\mathbf{H}_2^\top, \tag{1}$$

where $\mathbf{e}_j$ (resp. $\mathbf{H}_j$) is the vector obtained from $\mathbf{e}$ (resp. the matrix obtained from $\mathbf{H}$) by only keeping the coordinates (resp. the columns) indexed by I_j.

Equation (1) can be rewritten as

$$\mathbf{0} = \mathbf{e}\mathbf{H}^\top - \mathbf{s} = \mathbf{e}_1\mathbf{H}_1^\top + \mathbf{e}_2\mathbf{H}_2^\top - \mathbf{s}$$

i.e.,

$$\mathbf{e}_1\mathbf{H}_1^\top = \mathbf{s} - \mathbf{e}_2\mathbf{H}_2^\top \tag{2}$$

and Dumer's bet is that the support of the error $\mathbf{e}$ spreads evenly over $\mathcal{I}_j$. In other words, we want to find vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ of length $n/2$ and Hamming weight $w/2$.

Therefore, for each partition (I_1, I_2) of size $n/2$, we build the two lists

$$\begin{aligned} \mathcal{L}_1 &\stackrel{\text{def}}{=} \left\{ \mathbf{x}_1\mathbf{H}_1 \colon \mathbf{x}_1 \in \mathbb{F}_2^{n/2}, |\mathbf{x}_1| = w/2 \right\}, \\ \mathcal{L}_2 &\stackrel{\text{def}}{=} \left\{ \mathbf{s} - \mathbf{x}_2\mathbf{H}_2 \colon \mathbf{x}_2 \in \mathbb{F}_2^{n/2}, |\mathbf{x}_2| = w/2 \right\}, \end{aligned} \tag{3}$$

and we try to find collisions, *i.e.*, elements in the intersection $\mathcal{L}_1 \cap \mathcal{L}_2$.

Algorithm 2: Dumer algorithm

Input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k}$ and w that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$ for some $\mathbf{e}$ of weight w.
Output: A list of solutions $\mathbf{x}$ with $|\mathbf{x}| = w$ and $\mathbf{x}\mathbf{H}^\top = \mathbf{s}$.

```
1  Pick uniformly at random a partition I_1 ⊔ I_2 of {1,...,n}, with parts of size n/2
2  Build the lists L_1 and L_2 defined in Equation (3)
3  if L_1 ∩ L_2 = ∅ then
4      Go back to Step 1.
5  else
6      L ← ∅
7      foreach (x_1, x_2) corresponding to an element in L_1 ∩ L_2 do
8          Set x such that x_{I_1} = x_1 and x_{I_2} = x_2
9          L ← L ∪ {x}
10     return L
```

(Q2) Could you imagine a way to represent $\mathcal{L}_1$ and $\mathcal{L}_2$ to easily find collisions?
Hint: Use hash tables (or dictionnaries, in Python/SageMath)!

In Python/SageMath, the key of a dictionnary should be *immutable*, *i.e.*, it should not be allowed to modify it once it is created.
For example:

```
sage: d = {} # Create an empty hash table
sage: x = vector(GF(2), [0, 1]) # some binary vector;
                                # mutable.
sage: x
(0, 1)
sage: d[x] = 0
[...]
TypeError: mutable vectors are unhashable
sage: x.set_immutable() # Freeze it once and for all
sage: d[x] = 0
sage: d
{(0, 1): 0}
```

(Q3) Implement Dumer algorithm and test it on small examples.

(Q4) Check that your algorithm returns about $\frac{\binom{n/2}{w/2}^2}{2^{n-k}}$ solutions.

3 The Best of Both Worlds: First Information Set Decoding Algorithms

Dumer algorithm has a quadratic advantage over exhaustive search for decoding high rate codes at the Gilbert-Varshamov distance. However, it also returns *a list* of solutions, ... of size about the same as its time complexity! In other words, it finds solutions in *constant amortized time*. This is the key idea of *Information Set Decoding* algorithms (ISDs).

Indeed, let $I \subset \{1, \ldots, n$ be an information set of the code $\mathcal{C}$, of size k. Instead of asking the candidate solutions $\mathbf{x}$ to have no error on I as in Prange algorithm, we will relax this condition and allow a small number p of errors, but on a larger set $J \supset I$, of size $k + \ell$ for some parameters p and ℓ:

$$|\mathbf{x}_J| = p \quad \text{where } |J| = k + \ell \text{ and } J \supset I. \tag{4}$$

Note that there are very few constraints on those two parameters, we only ask

$$0 \leq \ell \leq n-k \quad \text{and} \quad p \leq \min\{k+\ell, w\}. \tag{5}$$

(Q5) How can we efficiently check that J contains an information set of $\mathcal{C}$?

(Q6) Let $\mathcal{C}_J = \{\mathbf{c}_J : \mathbf{c} \in \mathcal{C}\} \subset \mathbb{F}_2^{k+\ell}$ be obtained from $\mathcal{C}$ by only keeping the coordinates indexed by J. Show that if J contains an information set, then $\mathcal{C}_J$ is a code of length $k+\ell$ and dimension k.

We can then solve a smaller decoding problem of length $k+\ell$, dimension k (and therefore rate $1 - \frac{\ell}{k+\ell}$), and decoding distance p. Since this new code has rate close to one, we can efficiently make use of Dumer algorithm to recover a list of solutions of this smaller problem. However, we need to compute a parity-check matrix of this *punctured* code $\mathcal{C}_J$. Let $J' \overset{\text{def}}{=} \{1, \ldots, n\} \setminus J$ be the complement set of J. It has size $n-k-\ell$.

(Q7) Let $\mathbf{H}$ be a parity-check matrix of the code $\mathcal{C}$. Let $\mathbf{H}_J \in \mathbb{F}_2^{(n-k)\times(k+\ell)}$ and $\mathbf{H}_{J'} \in \mathbb{F}_2^{(n-k)\times(n-k-\ell)}$ be the submatrices obtained from $\mathbf{H}$ by keeping only the columns indexed by J (resp. J').

(a) Show that $\mathbf{H}_{J'}$ has full rank, *i.e.*, that it has rank $n-k-\ell$.

(b) Let $\mathbf{S} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ be a *non-singular* matrix such that

$$\mathbf{S}\mathbf{H}_{J'} = \begin{pmatrix} I_{n-k-\ell} \\ 0_{\ell\times(n-k-\ell)} \end{pmatrix}$$

and write

$$\mathbf{S}\mathbf{H}_J = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{pmatrix}.$$

Show that $\mathbf{H}_2$ is a parity-check matrix of the code $\mathcal{C}_J$.

(c) How can we compute such a matrix $\mathbf{S}$?

(Q8) Let $\mathbf{s}_1 \in \mathbb{F}_2^{n-k-\ell}$ and $\mathbf{s}_2 \in \mathbb{F}_2^{\ell}$ such that $\mathbf{s}\mathbf{S}^\top = \begin{pmatrix} \mathbf{s}_1 & \mathbf{s}_2 \end{pmatrix}$. Let $\mathbf{x}_2 \in \mathbb{F}_q^{k+\ell}$ be a solution of weight p of $\mathbf{x}_2\mathbf{H}_2^\top = \mathbf{s}_2$. Let $\mathbf{x} \in \mathbb{F}_2^n$ be a solution of the linear system $\mathbf{x}\mathbf{H}^\top = \mathbf{s}$ such that $\mathbf{x}_J = \mathbf{x}_2$.

(a) What should be the value of $\mathbf{x}_{J'}$?

(b) Conclude.

All in all, this yields the following algorithm

Algorithm 3: ISD algorithm using Dumer as a subroutine

Input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k}$, w and parameters p, ℓ satisfying Equation (5) and such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$ for some $\mathbf{e}$ of weight w.

Output: $\mathbf{x}$ with $|\mathbf{x}| = w$ and $\mathbf{x}\mathbf{H}^\top = \mathbf{s}$.

1 Pick uniformly at random a set $J \subset \{1, \dots, n\}$ of size $k + \ell$, and let $I \stackrel{\text{def}}{=} \{1, \dots, n\} \setminus J$.

2 **if** J does not contain an information set (Condition (Q5)) **then**

3   Goto Step 1

4 $\mathbf{E}, \mathbf{S} \leftarrow$ GaussianElimination$(\mathbf{H}_I)$

5 Define $\mathbf{H}_1 \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}, \mathbf{H}_2 \in \mathbb{F}_2^{\ell\times(k+\ell)}$ such that $\mathbf{S}\mathbf{H}_J = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{pmatrix}$

6 Define $\mathbf{s}_1 \in \mathbb{F}_2^{n-k-\ell}, \mathbf{s}_2 \in \mathbb{F}_2^{\ell}$ such that $\mathbf{s}\mathbf{S}^\top = \begin{pmatrix} \mathbf{s}_1 & \mathbf{s}_2 \end{pmatrix}$

7 Using Dumer Algorithm (2), compute a list of partial solutions

$$\mathcal{L} \subset \left\{ \mathbf{x}_2 \in \mathbb{F}_2^{k+\ell} \colon \mathbf{x}_2\mathbf{H}_2^\top = \mathbf{s}_2 \text{ and } |\mathbf{x}_J| = p \right\}$$

8 **foreach** $\mathbf{x}_2 \in \mathcal{L}$ **do**

9   Let $\mathbf{x} \in \mathbb{F}_2^n$ be a solution to the linear system $\mathbf{x}\mathbf{H}^\top = \mathbf{s}$ such that $\mathbf{x}_J = \mathbf{x}_2$.

10   **if** $|\mathbf{x}| = w$ **then**

11   **return x**

12   Go back to Step 1.

9. Implement Algorithm 3 in SageMath, and use it to break as many challenges as you can!

   Remark: You may want to play with different choices of parameters p and ℓ.