CRYPTO READING GROUP
HAMMING QUASI CYCLIC (HQC)

Spring 2026

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Contents

About

This document contains a detailed plan for the code-based cryptography study group.

Code-Based Cryptography

This semester, the reading group will focus on *code-based cryptography*. The idea is to explore a topic that lies somewhat outside the usual comfort zone of most of us, and to use the reading group as an opportunity to learn together.

Our main objective will be to build enough background to understand the design, security, and implementation aspects of the HQC cryptosystem. More broadly, the reading group will also serve as an introduction to some of the central ideas and techniques in code-based cryptography.

Structure

Each week, two participants will be assigned to prepare the material for that session together. The goal is to encourage collaboration, make the preparation more manageable, and allow for discussion before the presentation itself.

The speakers may then decide how they would like to present the material. For instance, they may choose to split the talk into two parts and each present one half, or they may decide that only one of them will deliver the presentation while both contribute to the preparation. Some coordination with the speakers from previous and following weeks may occasionally be helpful, but we expect this to remain fairly light.

Useful Information

Talks. The talks will be held in the **Mathematics & Computer Science (MC)** building at the University of Waterloo, in room **MC 5047**, located on the fifth floor.

Lunches. We are also planning to organize group lunches every other week, alternating between the lounge and the plaza.

Organizers

Leonardo Colò	University of Waterloo
Seunghoon Lee	University of Waterloo
Bruno Sterner	University of Waterloo

Schedule

Hamming Quasi-Cyclic

	Spring 2026		
May 15, 2026	1	**Bruno Sterner** **Seunghoon Lee**	Introduction to coding theory
May 22, 2026	2	**Maher Mamah** **Elle Wen**	Introduction to code based cryptography
May 22, 2026	**Lunch in the Plaza**		
May 29, 2026	3	**Pranshu Kumar** **John Premkumar** **Karaneh Keypoor**	Quasi-cyclic codes
June 1, 2026	4	**Roman Langrehr** **Sam Jaques**	Information Set Decoding
June 1, 2026	**Lunch in the Lounge**		
June 12, 2026	5	**Camryn Steckel**	Decoding for quasi-cyclic codes
June 19, 2026	6	**Mojtaba Fadavi** **Anna Henderson**	HQC PKE/KEM
June 19, 2026	**Lunch in the Plaza**		
June 26, 2026	7	**Bruce Xu** **Maggie Simmons**	HQC implementation/optimization
July 3, 2026	8	**Speaker 15** **Speaker 16**	Optional
July 3, 2026	**End of the term lunch**		

Plan

Introduction to Coding Theory I

Seunghoon Lee & Bruno Sterner 1

This first session introduces the basic language of coding theory and explains why coding-theoretic objects became relevant to cryptography. We will discuss the main definitions and parameters of codes, present linear codes as the fundamental setting of the theory, and study some classical examples such as Reed–Solomon codes and Goppa codes. The goal is to provide the algebraic and combinatorial background needed for the rest of the reading group.
References: [§2.1-2.2, 12] and [9]

Introduction to Code-Based Cryptography

Maher Mamah & Elle Wen 2

This session serves as an introduction to code-based cryptography. We will explain the decoding problem and its central role as a hardness assumption, then present the general McEliece framework and its security intuition. We will also give a first overview of the Goppa-code instantiation, with the aim of understanding how structured codes can be used to build public-key encryption schemes.
References: [§3.1, 12]

Quasi-Cyclic Codes

Pranshu Kumar & John Premkumar & Karaneh Keypoor 3

This session is devoted to quasi-cyclic codes, one of the main structured code families used in modern code-based cryptography. We will introduce their definition and main properties, and explain why their additional algebraic structure is both useful for efficiency and delicate from a security perspective. This week will provide the background needed to understand HQC and related constructions.
References: [7] and [§3.4, 12]

Information Set Decoding

Roman Langrehr & Sam Jaques 4

In this session, we study information set decoding (ISD), one of the main generic approaches for attacking code-based cryptosystems. We will present the basic ideas behind Prange's algorithm and Stern's algorithm, together with the general philosophy of decoding attacks in the random-code setting. The aim is to understand both the algorithmic framework and its importance in concrete security estimates.
References: [2] and [§5.3, 12]

Decoding for Quasi-Cyclic Codes

Camryn Steckel & 5

This session focuses on decoding questions specific to quasi-cyclic codes. We will discuss syndrome decoding in the quasi-cyclic setting and compare generic ISD methods with approaches that exploit additional structure. The goal is to better understand the tension between efficiency and security, and to prepare the ground for the study of the HQC scheme.
References: [§6.3, 4], [§3, 6], and [§5, 10]

HQC PKE/KEM

Mojtaba Fadavi & Anna Henderson 6

This session is devoted to the HQC cryptosystem itself, in both its public-key encryption and key-encapsulation forms. We will explain how the scheme works, describe its main components and design choices, and discuss the corresponding security analysis, including comments on the post-quantum setting. By this stage, the reading group should have enough background to appreciate both the structure and the rationale of HQC.
References: [1] and [4]

HQC Implementation and Optimisation

Bruce Xu & Maggie Simmons 7

This week will cover the implementation and optimization of key sub-routines within HQC. We will begin by examining the implementation of Reed-Solomon decoding within HQC, which includes the BCH-view of syndromes, weighted Newton's identity, the Berlekamp-Massey algorithm, and more. We will also discuss high-performance polynomial multiplication via the Karatsuba algorithm and hardware optimization.
References: [3] and [4]

Optional Session: Further Directions in Code-Based Cryptography

2 Speakers 8

This optional final session may be used to explore further directions in code-based cryptography. Possible topics include code-based signature schemes currently submitted to NIST, such as CROSS or LESS, or a more advanced topic building on the study of HQC and quasi-cyclic codes. Exact content to be adjusted.
References: [5] and [11]

Week 1

Introduction to coding theory

Talk by Seunghoon and Bruno.

Motivation

Coding theory studies how to encode messages so that they can be recovered even after errors occur during transmission or storage. Informally, we start with a message m, add redundancy, and obtain an encoded message $m' = m + (\text{redundancy})$. If some errors occur, the redundancy may allow us to recover the original message.

$$m' + (\text{errors}) \longmapsto m.$$

One of the classical motivations comes from the work of Richard Hamming. Coding theory is also an important background for code-based cryptography, one of the main families of post-quantum cryptography.

Linear Codes

Let q be a prime power, and let $\mathbb{F}_q$ be the finite field with q elements.

Definition ($[n,k]$-linear code)**.** *Let $k, n \in \mathbb{Z}_{>0}$ with $k \leq n$. An $[n,k]$-*linear code *over $\mathbb{F}_q$ is a k-dimensional linear subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$. An element $\mathbf{c} \in \mathcal{C}$ is called a codeword. The rate of $\mathcal{C}$ is $R = \frac{k}{n}$.*

Thus, n is the length of the code, while k is the dimension, or the number of information symbols (message).

Example (Repetition code)**.** *The binary repetition code of length 3 is $\mathcal{C} = \{000, 111\} \subseteq \mathbb{F}_2^3$. This is a $[3,1]$-linear code. It encodes one bit $m \in \mathbb{F}_2$ as $m \longmapsto (m,m,m)$.*

Generator and Parity-Check Matrices

Definition (Generator matrix)**.** *A matrix $G \in \mathbb{F}_q^{k \times n}$ is a generator matrix of an $[n,k]$-linear code $\mathcal{C}$ if $\mathcal{C} = \{\mathbf{x}G : \mathbf{x} \in \mathbb{F}_q^k\} = \langle G \rangle$, where $\langle G \rangle$ denotes the row span of G.*

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

Example. *For the binary repetition code $\mathcal{C} = \{000, 111\}$, a generator matrix is $G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \in \mathbb{F}_2^{1 \times 3}$. For $m \in \mathbb{F}_2$, the corresponding codeword is $\mathbf{c} = mG = (m,m,m)$.*

Definition (Parity-check matrix). *A matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ is a parity-check matrix of $\mathcal{C}$ if $\mathcal{C} = \{\mathbf{y} \in \mathbb{F}_q^n : H\mathbf{y}^T = 0\}$.*

Equivalently, H gives linear equations that characterize membership in $\mathcal{C}$.

Example. *For the repetition code $\mathcal{C} = \{000, 111\}$, a vector $\mathbf{y} = (y_1, y_2, y_3)$ is a codeword precisely when $y_1 + y_2 = 0$ and $y_1 + y_3 = 0$. Hence, we may take $H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$. One can check that $GH^T = 0$. Note that the parity-check matrix may not be unique. For our example, $H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ is also a parity-check matrix, which comes from the equations $y_1 + y_3 = 0$ and $y_2 + y_3 = 0$.*

Definition (Syndrome). *For any received word $\mathbf{r} \in \mathbb{F}_q^n$, the vector $H\mathbf{r}^T$ is called the syndrome of $\mathbf{r}$.*

If $\mathbf{r} = \mathbf{c} + \mathbf{e}$ for some codeword $\mathbf{c} \in \mathcal{C}$ and error vector $\mathbf{e}$, then $H\mathbf{r}^T = H(\mathbf{c} + \mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = H\mathbf{e}^T$. Thus, the syndrome depends only on the error vector $\mathbf{e}$, not on the transmitted codeword $\mathbf{c}$.

Example. *For the repetition code with $H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$, consider the single-bit errors $\mathbf{e}_1 = 100, \mathbf{e}_2 = 010, \mathbf{e}_3 = 001$. Their syndromes are $H\mathbf{e}_1^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, H\mathbf{e}_2^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, H\mathbf{e}_3^T = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Since these syndromes are distinct, the syndrome tells us the location of a single error.*

Hamming Weight and Hamming Distance

Definition (Hamming weight). *The Hamming weight of $\mathbf{x} \in \mathbb{F}_q^n$ is $\mathrm{wt}(\mathbf{x}) = |\{i \in [n] : x_i \neq 0\}|$.*

Definition (Hamming distance). *The Hamming distance between $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) = |\{i \in [n] : x_i \neq y_i\}|$.*

Definition (Minimum Hamming distance). *The minimum Hamming distance of a code $\mathcal{C}$ is $d_{\mathrm{H}}(\mathcal{C}) = \min\{d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\}$.*

Example. *For the repetition code $\mathcal{C} = \{000, 111\}$, we have $d_{\mathrm{H}}(\mathcal{C}) = d_{\mathrm{H}}(000, 111) = 3$.*

Exercise. *Let $\mathcal{C}$ be a linear code. Show that $d_{\mathrm{H}}(\mathcal{C}) = \min\{\mathrm{wt}(\mathbf{x}) : \mathbf{x} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{0}\}$.*

Proof. Since $\mathcal{C}$ is linear, if $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, then $\mathbf{x} - \mathbf{y} \in \mathcal{C}$. Also, $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x} - \mathbf{y})$. Therefore, $d_{\mathrm{H}}(\mathcal{C}) = \min\{d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\} = \min\{\mathrm{wt}(\mathbf{x} - \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\}$. As $\mathbf{x} - \mathbf{y}$ ranges over nonzero codewords of $\mathcal{C}$, this becomes $d_{\mathrm{H}}(\mathcal{C}) = \min\{\mathrm{wt}(\mathbf{z}) : \mathbf{z} \in \mathcal{C},\ \mathbf{z} \neq \mathbf{0}\}$. □

Example. *The previous statement can fail if $\mathcal{C}$ is not linear. For example, let $\mathcal{C} = \{000, 111, 110\} \subseteq \mathbb{F}_2^3$. This code is not linear, since $111 + 110 = 001 \notin \mathcal{C}$. Here, $d_{\mathrm{H}}(\mathcal{C}) = d_{\mathrm{H}}(111, 110) = 1$, but $\min\{\mathrm{wt}(\mathbf{x}) : \mathbf{x} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{0}\} = \mathrm{wt}(110) = 2$.*

Error Correction

Definition (Correction up to t errors)**.** *We say that a code $\mathcal{C}$ can correct up to t errors if, for every received word $\mathbf{r} \in \mathbb{F}_q^n$ with $d_{\mathrm{H}}(\mathbf{r}, \mathcal{C}) \leq t$, there exists a unique codeword $\mathbf{c} \in \mathcal{C}$ such that $d_{\mathrm{H}}(\mathbf{r}, \mathbf{c}) \leq t$.*

Here, $d_{\mathrm{H}}(\mathbf{r}, \mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}} d_{\mathrm{H}}(\mathbf{r}, \mathbf{c})$. A decoding algorithm is then expected to map such a received word $\mathbf{r}$ to the unique closest codeword $\mathbf{c}$.

Exercise. *Let $\mathcal{C}$ be a linear code of length n over $\mathbb{F}_q$ with $d_{\mathrm{H}}(\mathcal{C}) = d$. Show that $\mathcal{C}$ can correct up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors.*

Proof. First, we show existence. Suppose $\mathbf{r}$ is obtained from some codeword $\mathbf{c} \in \mathcal{C}$ by adding an error vector $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) \leq t$. Then $\mathbf{r} = \mathbf{c} + \mathbf{e}$. Thus, $d_{\mathrm{H}}(\mathbf{r}, \mathbf{c}) = \mathrm{wt}(\mathbf{e}) \leq t$. So there exists at least one codeword within distance t of $\mathbf{r}$.

Next, we show uniqueness. Suppose $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ are two codewords such that $d_{\mathrm{H}}(\mathbf{r}, \mathbf{c}_1) \leq t$ and $d_{\mathrm{H}}(\mathbf{r}, \mathbf{c}_2) \leq t$. By the triangle inequality, $d_{\mathrm{H}}(\mathbf{c}_1, \mathbf{c}_2) \leq d_{\mathrm{H}}(\mathbf{c}_1, \mathbf{r}) + d_{\mathrm{H}}(\mathbf{r}, \mathbf{c}_2) \leq 2t$. Since $t = \lfloor (d-1)/2 \rfloor$, we have $2t \leq d - 1$. Therefore, $d_{\mathrm{H}}(\mathbf{c}_1, \mathbf{c}_2) \leq d - 1$. But distinct codewords in $\mathcal{C}$ have distance at least d. Hence $\mathbf{c}_1$ and $\mathbf{c}_2$ cannot be distinct. Therefore, $\mathbf{c}_1 = \mathbf{c}_2$. □

$[n, k, d]$-Linear Codes and the Singleton Bound

Definition. *An $[n, k, d]$-linear code is an $[n, k]$-linear code $\mathcal{C}$ such that $d = d_{\mathrm{H}}(\mathcal{C})$.*

There is a tradeoff between rate and minimum distance. We want the rate $R = \frac{k}{n}$ to be large, but we also want the relative distance $\delta = \frac{d}{n}$ to be large.

Theorem (Singleton bound)**.** *Let $\mathcal{C}$ be an $[n, k]$-linear code over $\mathbb{F}_q$ with minimum distance $d = d_{\mathrm{H}}(\mathcal{C})$. Then $d \leq n - k + 1$.*

Proof. Define a map $\varphi : \mathcal{C} \to \mathbb{F}_q^{n-d+1}$ by deleting the last $d-1$ coordinates: $\varphi(x_1, \ldots, x_n) = (x_1, \ldots, x_{n-d+1})$. We claim that φ is injective. Suppose not. Then there exist distinct codewords $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ such that $\varphi(\mathbf{c}_1) = \varphi(\mathbf{c}_2)$. This means that $\mathbf{c}_1$ and $\mathbf{c}_2$ agree on the first $n - d + 1$ coordinates, so they can differ only in the remaining $d - 1$ coordinates. Hence $d_{\mathrm{H}}(\mathbf{c}_1, \mathbf{c}_2) \leq d - 1$, which contradicts the definition of d.

Therefore, φ is injective. Hence $|\mathcal{C}| \leq |\mathbb{F}_q^{n-d+1}| = q^{n-d+1}$. Since $\mathcal{C}$ has dimension k, we also have $|\mathcal{C}| = q^k$. Therefore, $q^k \leq q^{n-d+1}$, so $k \leq n - d + 1$. Rearranging gives $d \leq n - k + 1$. □

Intuition. The minimum distance d means that any two distinct codewords differ in at least d positions. Therefore, even after deleting $d - 1$ positions, two distinct codewords cannot become equal. This is the key reason why the deletion map in the proof is injective.

Definition (Maximum distance separable code)**.** *A code $\mathcal{C}$ satisfying $d = n - k + 1$ is called a maximum distance separable code, or MDS code.*

MDS codes achieve the maximum possible minimum distance for fixed n and k. Therefore, they can correct the maximum possible number of errors for those parameters. Reed–Solomon codes are a central example of MDS codes.

Hamming Balls and the Gilbert–Varshamov Bound

For $r \geq 0$, define the Hamming ball of radius r in $\mathbb{F}_q^n$ by $B_{\mathrm{H}}(r, n, q) = \left\{\mathbf{x} \in \mathbb{F}_q^n : \mathrm{wt}(\mathbf{x}) \leq r\right\}$. Its size is $|B_{\mathrm{H}}(r, n, q)| = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i$.

The Gilbert–Varshamov bound gives a sufficient condition for the existence of good linear codes.

Theorem (Gilbert–Varshamov bound, informal form)**.** *If $|B_{\mathrm{H}}(d-2, n-1, q)| < q^{n-k}$, then there exists an $[n, k]$-linear code over $\mathbb{F}_q$ with minimum distance at least d.*

The main message is that good linear codes exist. For suitable parameters, one can have both nontrivial rate and nontrivial relative minimum distance. This is part of the background reason why random-looking good-rate linear codes are meaningful objects in code-based cryptography.

Nontrivial Rate and Nontrivial Minimum Distance

For a family of $[n, k, d]$-linear codes, nontrivial rate and nontrivial relative distance mean that there exist constants $R_0 > 0$ and $\delta_0 > 0$ such that $R = \frac{k}{n} \geq R_0$ and $\delta = \frac{d}{n} \geq \delta_0$. That is, both k and d are proportional to n.

Example (Repetition codes)**.** *A binary repetition code of length n has parameters $[n, 1, n]$. Thus its relative distance is 1, but its rate tends to 0 as n grows.*

Example (Hamming codes)**.** *A binary Hamming code has parameters $[2^r - 1,\ 2^r - r - 1,\ 3]$. Thus its rate tends to 1, but its relative distance tends to 0 as r grows.*

Reed-Solomon Codes

We have already seen one example of an MDS code in the $[n, 1]$ repetition code. This can be checked directly as its minimum Hamming distance is n. However this is has a low rate and a lot of reducancy is needed for correcting errors. The briefly mentioned Hamming code has a significantly better rate but is not an MDS code. We will give an overview of the *Reed Solomon code* which is an MDS code that has a very good rate.

Definition. *Fixed some distinct and non-zero $\alpha_1, \cdots, \alpha_n \in \mathbb{F}_q$ and write $\alpha = (\alpha_1, \cdots, \alpha_n)$. For an integer $k \leq n$, the* Reed Solomon code, $\mathsf{RS}_{n,k}(\alpha)$, *is defined as*

$$\mathsf{RS}_{n,k}(\alpha) \coloneqq \{(f(\alpha_1), \cdots, f(\alpha_n)) : f \in \mathbb{F}_q[x] \text{ with } \deg(f) \leq k-1\}.$$

Lemma. $\mathsf{RS}_{n,k}(\alpha)$ *is an (n,k)-linear code with the following generator and parity check matrices:*

Proof (sketch).

1. The linearity follows from the fact that $\mathbb{F}_q[x]$ is a ring.
2. For any $f(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1} \in \mathbb{F}_q[x]$, the codeword $c_f = (f(\alpha_1), \cdots, f(\alpha_n))$ can be written as
$$c_f = (a_0\ a_1\ \cdots\ a_{k-1})G_{\mathsf{RS}}.$$
3. More involved and omitted (can be done by looking at the dual code). □

One can transmit a message $m = (m_0, \cdots, m_{k-1}) \in (\mathbb{F}_q)^k$ as a codeword

$$c_m = (f_m(\alpha_1), \cdots, f_m(\alpha_n)) \in \mathsf{RS}_{n,k}(\alpha),$$

where $f_m(x) = m_0 + m_1 x + \cdots + m_{k-1}x^{k-1} \in \mathbb{F}_q[x]$. The extra $n-k$ entries in this codeword compared to the message acts as the redundancy.

Question: Given some transmition $y = c_f \oplus e$ of a codeword $c_f \in \mathsf{RS}_{n,k}(\alpha)$, how many errors in c_f can we correct? Are there efficient decoding algorithms to actually correct such errors.

Proposition. $\mathsf{RS}_{n,k}(\alpha)$ *is an MDS code, i.e.* $d = d_H(\mathsf{RS}_{n,k}(\alpha)) = n - k + 1$

Proof (sketch). Reformulate the minimal Hamming distance to

$$d_H(\mathcal{C}) = \max\left\{ d' \in \mathbb{N} : \begin{array}{c}\text{any } (d'-1) \text{ columns of the parity-check} \\ \text{matrix are linearly independent}\end{array} \right\}$$

(See Roth's book [9, Theorem 2.2]).

Any $(n-k) \times (n-k)$ submatrix of H_{RS} is non-singular (since it has a Vandemonde form). So any $(n-k)$ columns of H_{RS} are linearly independent and hence $d \geq n - k + 1$. With the singleton bound we get an equality: $d = n - k + 1$. □

So one can correct at most $\lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2$ errors (which is maximal among all $[n,k]$-linear codes. Remarkably the answer to the second part of the above question is yes – with this maximal number of corrections.

Theorem (informal)**.** *There are efficient decoding algorithms for* $\mathsf{RS}_{n,k}[\alpha]$ *that can correct* $\lfloor (n-k)/2$ *errors.*

One of these algorithms is based on a truncated version of the extended Euclidean algorithm.

Example. *The Reed Solomon code with $n = 255$ and $k = 223$ commonly used in many applications (e.g. communication between NASA and Voyager missions). It can correct up to 16 errors in a transmitted codeword.*

The q-ary entropy function

The q-ary entropy function is $h_q(x) = x \log_q(q-1) - x \log_q x - (1-x)\log_q(1-x)$. For binary codes, the Gilbert–Varshamov bound gives the asymptotic guarantee $R \gtrsim 1 - h_2(\delta)$.

Some rough tradeoffs are: $R \approx 0.8 \implies \delta \approx 0.03$, $R \approx 0.5 \implies \delta \approx 0.11$, and $R \approx 0.28 \implies \delta \approx 0.20$.

For McEliece-type parameter examples, one often sees roughly $R \approx 0.7$–0.8, $t/n \approx 0.015$–0.02, and $d/n \approx 0.03$–0.04. Here t is the number of correctable errors, and t/n is the error rate.

Week 2

Introduction to code-based cryptography

Talk by Maher and Elle.

Decoding Problems

Let $C \subseteq \mathbb{F}_q^n$ be an $[n, k]$ linear code with parity-check matrix

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

Thus

$$C = \{c \in \mathbb{F}_q^n : Hc^\top = 0\}.$$

Equivalently, if G is a generator matrix for C, then

$$C = \{c : c = m^\top G,\ m \in \mathbb{F}_q^k\}.$$

For a received word $y = c + e$, where $c \in C$ and $\mathrm{wt}(e) \leq t$, its syndrome is

$$Hy^\top = H(c + e)^\top = Hc^\top + He^\top = He^\top.$$

Hence the syndrome depends only on the error vector. Decoding can therefore be written as the syndrome decoding problem.

Decoding Problem (DP) Given G and y, find m and $e \in \mathbb{F}_q^n$ such that $y = m^\top G + e,$ and $\mathrm{wt}(e) \leq t.$

Syndrome Decoding Problem (SDP) Given H and s, find $e \in \mathbb{F}_q^n$ such that $s = He^\top,$ and $\mathrm{wt}(e) \leq t.$

Theorem (Theorem 183 of survey paper)**.** *DP and SDP are equivalent.*

For a random-looking linear code, this problem is assumed to be computationally hard.

Remark. *The hardness of syndrome decoding comes from the sparsity constraint, not from the linear equation itself. Without*

$$\mathrm{wt}(e) \leq t,$$

the equation

$$He^\top = s$$

is solvable by linear algebra. The difficulty is to find a solution with small Hamming weight.

If t is small, decoding is easier. If t is large, decoding is harder, because there are many more possible error vectors. Over $\mathbb{F}_2$, the condition $\mathrm{wt}(e) = t$ means that the error is supported on exactly t coordinates. Thus one must identify the error support among

$$\binom{n}{t}$$

possible subsets. For random-looking H, there is no exploitable structure revealing this support, so generic attacks remain exponential-time combinatorial searches.

Let

$$H \in \mathbb{F}_q^{(n-k)\times n}, \qquad s \in \mathbb{F}_q^{n-k}, \qquad t \in \mathbb{N}.$$

The *Decisional Syndrome Decoding Problem* asks whether there exists $e \in \mathbb{F}_q^n$ such that

$$He^\top = s, \qquad \mathrm{wt}_H(e) \le t.$$

Berlekamp, McEliece, and van Tilborg proved that binary SDP in the Hamming metric is NP-complete, via a reduction from 3-Dimensional Matching. Strictly, **NP-hardness and NP-completeness** apply to decision problems. The search version asks to find such an e, but a polynomial-time search algorithm would also solve the decision version by verification. Thus the computational SDP is often informally referred to as NP-hard.

The best known generic attacks on SDP are *information-set decoding* (ISD) algorithms. ISD solves SDP probabilistically by combinatorial search. Starting from Prange's algorithm, several refinements have been proposed, including Dumer, MMT, BJMM, May–Ozerov, and Both–May. Their costs remain exponential, typically written as

$$2^{\alpha_{R,\tau} n + o(n)}, \qquad R = k/n \text{ (code rate)}, \qquad \tau = t/n \text{ (error rate)}.$$

Unique decoding is usually guaranteed only up to half the minimum distance:

$$\left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor.$$

Full-distance decoding refers to the asymptotic regime where the error weight is large, on the scale of the typical minimum distance of a random code. If t becomes too large, there may be many solutions, so uniqueness may fail. Full-distance decoding is not about guaranteed unique decoding. It is used as a benchmark regime where the combinatorial search is very difficult for ISD-style algorithms.

In the full-distance decoding regime, Both–May achieves

$$2^{0.0885n + o(n)},$$

the smallest known asymptotic exponent among existing ISD variants in that setting.

Abstract McEliece PKE Scheme

The scheme consists of three algorithms:

$$\text{KeyGen}, \qquad \text{Enc}, \qquad \text{Dec}.$$

Key Generation

Choose a structured linear code with parity-check matrix

$$H_{\text{sec}} \in \mathbb{F}_q^{(n-k)\times n}$$

that admits efficient decoding up to t errors. Choose an invertible matrix

$$S \in \mathbb{F}_q^{(n-k)\times(n-k)}$$

and a permutation matrix

$$P \in \mathbb{F}_q^{n\times n}.$$

Set

$$H_{\text{pub}} = SH_{\text{sec}}P.$$

The public and secret keys are

$$\mathsf{pk} = H_{\text{pub}}, \qquad \mathsf{sk} = \big(H_{\text{sec}}, S, P, \mathsf{Decode}_{H_{\text{sec}}}\big).$$

Encryption

Represent the message as a vector

$$e \in \mathbb{F}_q^n, \qquad \mathrm{wt}(e) = t.$$

The ciphertext is its syndrome:

$$\mathsf{Enc}_{\mathsf{pk}}(e) = s = H_{\text{pub}}e^\top.$$

Decryption

Given s, compute

$$s' = S^{-1}s.$$

Since $s' = H_{\text{sec}}Pe^\top$, let $e'^\top = Pe^\top$. The vector e' has weight t, because P is a permutation. Use the secret decoder to recover e' from

$$s' = H_{\text{sec}}e'^\top.$$

Finally output

$$e^\top = P^{-1}e'^\top.$$

IND-CPA Issue of the Textbook Syndrome Formulation

The textbook syndrome formulation is not IND-CPA secure when the plaintext is encoded deterministically as a low-weight vector. Indeed, suppose the public key is

$$H_{\text{pub}} \in \mathbb{F}_q^{(n-k)\times n},$$

and encryption maps a message-encoding $e \in \mathbb{F}_q^n$, with $\mathrm{wt}(e) = t$, to the syndrome $s = H_{\text{pub}}e^\top$.

In the IND-CPA experiment, the adversary chooses two messages. Assume these messages are deterministically encoded as

$$e_1, e_2 \in \mathbb{F}_q^n, \qquad \mathrm{wt}(e_1) = \mathrm{wt}(e_2) = t.$$

The challenger samples $b \in \{1, 2\}$ and returns

$$s = H_{\mathrm{pub}} e_b^\top.$$

However, since H_{pub}, e_1, and e_2 are all known to the adversary, the adversary can compute

$$s_1 = H_{\mathrm{pub}} e_1^\top, \qquad s_2 = H_{\mathrm{pub}} e_2^\top.$$

Then the adversary simply compares the challenge ciphertext s with s_1 and s_2. If

$$s = s_1,$$

it outputs $b = 1$, and if

$$s = s_2,$$

it outputs $b = 2$. Thus the adversary distinguishes the challenge ciphertext with probability 1, except in the degenerate case where

$$H_{\mathrm{pub}} e_1^\top = H_{\mathrm{pub}} e_2^\top.$$

This attack does not require decoding the syndrome. It only uses the fact that the textbook map is deterministic: a fixed encoding e always gives the same syndrome $H_{\mathrm{pub}} e^\top$.

Random Padding in the Syndrome Formulation

In the textbook syndrome formulation, encryption is deterministic if a plaintext is encoded as a unique low-weight vector e:

$$s = H_{\mathrm{pub}} e^\top.$$

Thus an adversary can test candidate plaintexts by recomputing their syndromes.

Random padding removes this direct test. Instead of encoding a plaintext m as a unique vector, choose fresh randomness r and encode

$$(m, r) \longmapsto e_{m,r} \in \mathbb{F}_q^n, \qquad \mathrm{wt}(e_{m,r}) = t.$$

Encryption outputs

$$s = H_{\mathrm{pub}} e_{m,r}^\top.$$

To decrypt, one can use the decryption algorithm as before to obtain $e_{m,r}$ and then invert the encoding to obtain (m, r).

In the IND-CPA experiment, suppose the adversary chooses m_1, m_2, and the challenger encrypts m_1. Then

$$s = H_{\mathrm{pub}} e_{m_1,r}^\top$$

for fresh random r. The adversary can no longer compute a single candidate syndrome for m_1, because m_1 corresponds to many possible vectors

$$\{e_{m_1,r} : r \in \mathcal{R}\}.$$

Similarly, m_2 corresponds to the set

$$\{e_{m_2,r} : r \in \mathcal{R}\}.$$

Hence the old deterministic test

$$s \stackrel{?}{=} H_{\text{pub}} e_1^\top$$

is no longer available. To decide whether s came from m_1 or m_2, the adversary would need to find some r such that

$$s = H_{\text{pub}} e_{m_i,r}^\top,$$

or equivalently recover a valid low-weight preimage $e_{m_i,r}$ of the syndrome. This is again a syndrome decoding type problem. Thus random padding prevents the direct comparison attack: the same plaintext m may produce many possible syndromes, depending on the random padding r.

Next, we move on to treat the instantiation of McEliece using Binary Goppa codes, as originally proposed by McEliece himself in 1978.

Goppa Codes

Definition. *Fix parameters:*

- $m \geq 1$ *positive integer, prime* $q \geq 2$, $\mathbb{F}_{q^m}$ *finite field;*
- *Let* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$ *such that* α_i *are distinct,* $n \leq q^m$*;*
- *Pick* $g(x) \in \mathbb{F}_{q^m}[x]$ Goppa polynomial *such that* $g(\alpha_i) \neq 0 \, \forall \, i$ *and* $\deg g(x) \leq t$*.*

The Goppa code is defined as follows:

$$\Gamma(\boldsymbol{\alpha}, g) = \left\{ c \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \pmod{g(x)} \right\}.$$

With a straightforward calculation, one can easily check that

$$\frac{1}{x - \alpha_i} = -\frac{g(x) - g(\alpha_i)}{x - \alpha_i} \, g^{-1}(\alpha_i) \pmod{g(x)}.$$

Parity-Check Matrix H

Rewrite $g(x) = \sum_{i=0}^{t} g_i x^i, \quad g_i \in \mathbb{F}_{q^m}$.

$$\begin{aligned} \frac{g(x) - g(\alpha_i)}{x - \alpha_i} &= \frac{g_t(x^t - \alpha_i^t) + \cdots + g_1(x - \alpha_i) + g_0 \cdot 0}{x - \alpha_i} \\ &= g_t\big(x^{t-1} + \alpha_i x^{t-2} + \cdots + \alpha_i^{t-1}\big) + \cdots + g_2(x + \alpha_i) + g_1. \end{aligned}$$

Look at the coefficients of a codeword $\displaystyle\sum_{i=1}^{n} c_i \cdot \frac{g(x) - g(\alpha_i)}{x - \alpha_i} \, g^{-1}(\alpha_i)$:

$$
\begin{aligned}
x^{t-1}: &\quad g_t \cdot g^{-1}(\alpha_1)\, c_1 + \cdots + g_t \cdot g^{-1}(\alpha_n)\, c_n \\
&\vdots \\
x^{0}: &\quad \left(g_1 + \cdots + g_t \alpha_1^{t-1}\right) g^{-1}(\alpha_1)\, c_1 + \cdots + \left(g_1 + \cdots + g_t \alpha_n^{t-1}\right) g^{-1}(\alpha_n)\, c_n.
\end{aligned}
$$

Observe: $\deg(P) \leq t-1, \deg(g) \leq t \Rightarrow P$ *and also* $g(x)$ *divides* $P(x)$. *So polynomial* $P(x)$ *has* 0 *on its coefficients*

$$
c \in \Gamma(\boldsymbol{\alpha}, g) \iff \bar{H}\, c = 0 \quad \text{for } \bar{H} \in \mathbb{F}_{q^m}^{t \times n}
$$

$$
\mathbb{F}_{q^m}^{t \times n} \ni \bar{H} = \begin{pmatrix} g_t \cdot g^{-1}(\alpha_1) & \cdots & g_t \cdot g^{-1}(\alpha_n) \\ \vdots & & \vdots \\ \left(g_1 + \cdots + g_t \alpha_1^{t-1}\right) g^{-1}(\alpha_1) & \cdots & \left(g_1 + \cdots + g_t \alpha_n^{t-1}\right) g^{-1}(\alpha_n) \end{pmatrix}
$$

Applying a bijection via a fixed basis gives:

$$
\mathbb{F}_{q}^{tm \times n} \ni H = \begin{pmatrix} g^{-1}(\alpha_1) & \cdots & g^{-1}(\alpha_n) \\ \alpha_1\, g^{-1}(\alpha_1) & & \alpha_n\, g^{-1}(\alpha_n) \\ \vdots & & \vdots \\ \alpha_1^{t-1}\, g^{-1}(\alpha_1) & \cdots & \alpha_n^{t-1}\, g^{-1}(\alpha_n) \end{pmatrix}.
$$

Classic McEliece Key Generation

Set $q = 2$. Pick a monic irreducible $g(x) \in \mathbb{F}_{2^m}[x]$ of degree t, and n random distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{2^m}$. Convert to systematic form:

$$
\tilde{H}(g, \boldsymbol{\alpha}) \xrightarrow{\text{systematic form}} H = (I \mid T) \in \mathbb{F}_2^{tm \times n}, \qquad \boldsymbol{\alpha} \longrightarrow \boldsymbol{\alpha}'
$$

- **Secret key (sk):** $g(x),\ \boldsymbol{\alpha}'$
- **Public key (pk):** $T,\ t$ — assumes T is indistinguishable from a random binary matrix, and that the public key is useless for making decoding efficient.

$+$ Safe for large parameters (other GRS codes are broken).

$-$ Key size is too large: final-round pk size $= 261$ kilobytes (HQC is 2.2).

Decoding Goppa Codes

Received word $y = c + e$ with minimum distance $d \geq t + 1$ ($2t$ for binary), and

$$w(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor, \qquad B = \{i \mid e_i \neq 0\} \quad \text{(error positions)}.$$

The *syndrome* of y (which depends only on e) is:

$$S(x) = \sum_i \frac{y_i}{x - \alpha_i} = \sum_i \frac{e_i}{x - \alpha_i} \pmod{g(x)}.$$

Define two helper polynomials:

$$\text{error locator:} \quad \sigma(x) = \prod_{i \in B} (x - \alpha_i),$$

$$\text{error evaluator:} \quad \omega(x) = \sum_{i \in B} e_i \prod_{j \in B} (x - \alpha_j).$$

These are found via polynomial interpolation or Patterson Algorithm for binary Goppa codes (using Euclidean Algorithm). There is a polynomial-time procedure to correct t errors using degree-t polynomials.

Week 3

Quasi-cyclic codes

Talk by Pranshu, John, and Karaneh.

Motivation

So far, we've worked with linear codes, where the space of codewords is identified by the generator matrix. In cryptographic contexts, these generator matrices often become quite large, incurring significant communication costs. To reduce these communication costs, we try to add more structure to our codes, and explore their uses in cryptography.

Cyclic Codes

Let $\mathbb{F}_q$ denote the finite field with q elements, where q is a prime power, and let m, l, n be positive integers.

Definition (Cyclic Code)**.** *A linear code C of length n over $\mathbb{F}_q$ is called a* cyclic code *if it is invariant under right cyclic shift of codewords. That means, any codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in C$ implies another codeword $\mathbf{c}' = (c_{n-1}, c_0, c_1, \cdots, c_{n-2})$ is also in C.*

For notational convenience, we define T to be the shift-by-1 operator on a vector, and T^l to be the shift-by-l operator. So we have $T \cdot \mathbf{c} = \mathbf{c}'$ where $\mathbf{c}, \mathbf{c}'$ are defined in the above definition.

As a consequence of this added structure, cyclic codes correspond to ideals in polynomial rings, instead of just vector subspaces. Consider the map:

$$\varphi_c : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q[x]/ < x^n - 1 >$$
$$\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \longmapsto \mathbf{c}(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

Under this map, we get a nice algebraic representation of the cyclic shift as $\varphi_c(T \cdot \mathbf{c}) = x \cdot \varphi_c(\mathbf{c})$. Finally we have the following lemma characterizing the structure of cyclic codes:

Lemma. *There is a 1-to-1 correspondence between cyclic subspaces of $\mathbb{F}_q^n$ and ideals of $\mathbb{F}_q[x]/ < x^n - 1 >$.*

Due to this added structure in Cyclic codes, a single codeword implies $n - 1$ other codewords. So we can reduce the cost of communicating the generator matrix by up to a factor of n. As an example, for a codeword $\mathbf{c} = (c_0, c_1, \cdots, c_{n-1}) \in C$, the circulant matrix:

$$Circ(\mathbf{c}) = \begin{pmatrix} c_0, & c_1, & \cdots, & c_{n-1} \\ c_{n-1}, & c_0, & \cdots, & c_{n-2} \\ \vdots & \vdots & & \vdots \\ c_1, & c_2, & \cdots, & c_0 \end{pmatrix}$$

can act as a generator matrix for the code C.

Quasi-Cyclic Codes

Quasi-Cyclic (QC) codes are a generalization of cyclic codes.

Definition (Quasi-Cyclic Code)**.** *A linear code C of length ml over $\mathbb{F}_q$ is called a QC code of index l if it is invariant under shift of codewords by l positions and l is the minimal number with this property.*

Note that, if $l = 1$, then C is a cyclic code. If we view codewords of C as $m \times l$ arrays as follows

$$\mathbf{c} = \begin{pmatrix} c_{00} & \cdots & c_{0,l-1} \\ \vdots & & \vdots \\ c_{m-1,0} & \cdots & c_{m-1,l-1} \end{pmatrix},$$

then being invariant under shift by l units amounts to being closed under the row shift, where each row is moved downward one row, and the bottom row is moved to the top.

Grouping the coefficients of $\mathbf{c}$ together by rows, results in the following map:

$$\varphi_1 : \mathbb{F}_q^{ml} \longrightarrow \mathbb{F}_{q^l}^m$$

with a shift by l positions in the domain of the map corresponds to a shift by 1 position in the range: $\varphi_1(T \cdot \mathbf{c}) = T \cdot \varphi_1(\mathbf{c})$. This observation results in the following lemma:

Lemma. *Every QC code of index l can be viewed as a cyclic code of length l.*

For a ring $R = \mathbb{F}_q / < x^m - 1 >$, defining $\mathbf{c}_j(x) = c_{0,j} + c_{1,j}x + c_{2,j}x^2 + \cdots + c_{m-1,j}x^{m-1} \in R$ allows us to group group the coefficients of $\mathbf{c}$ column-wise and induces the following map:

$$\begin{aligned} \varphi_2 : \mathbb{F}_q^{ml} &\longrightarrow R^l = (\mathbb{F}_q / < x^m - 1 >)^l \\ \mathbf{c} &\longmapsto (\mathbf{c}_0(x), \mathbf{c}_1(x), \cdots, \mathbf{c}_{l-1}(x)) \end{aligned}$$

Under this map, we observe that a shift by l position on the input vector corresponds to multiplication by x on the output vector: $\varphi_2(T^l \cdot \mathbf{c}) = x \cdot \varphi_2(\mathbf{c})$. This maps allows us to characterize the structure of QC codes with the following lemma:

Lemma. *Every QC code of length ml and index l is a R-submodule of R^l where $R = \mathbb{F}_q/ < x^m - 1 >$.*

Tanner Graphs

Before introducing MDPC codes, we first introduce a useful graphical representation of binary linear codes called a *Tanner graph*. We restrict our discussion to binary linear codes over $\mathbb{F}_2$.

Definition (Tanner Graph)**.** *Let $H \in \mathbb{F}^{r\times n}$ be a parity-check matrix. The* Tanner graph *associated to H is a bipartite graph with two types of nodes:*

- *variable nodes $v_0, v_1, \ldots, v_{n-1}$ corresponding to the columns of H,*
- *check nodes $u_0, u_1, \ldots, u_{r-1}$ corresponding to the rows of H.*

There is an edge between variable node v_j and check node u_i if and only if $H_{i,j} = 1$

For example, consider a binary linear code with parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Then the Tanner graph has three variable nodes $v_0, \ldots, v_2$ and two check nodes u_0, u_1. The first row of H gives edges from u_0 to v_0 and v_1, the second row gives edges from u_1 to v_1 and v_2.

Since codewords satisfy

$$H\mathbf{c}^T = 0,$$

each row of H gives a parity-check equation. In the example above, for a codeword

$$\mathbf{c} = (c_0, c_1, c_2),$$

the first row gives the equation

$$c_0 + c_1 = 0 \mod 2,$$

and so on.

Thus, the edges incident to a check node correspond exactly to the variables that participate in the associated parity-check equation.

This graphical representation is useful because it gives a natural way to design decoding algorithms. For instance, in a very rough bit-flipping decoder, one computes the syndrome

$$\mathbf{s} = H\mathbf{r}^T,$$

where $\mathbf{r}$ is the received word. A nonzero syndrome coordinate $s_i = 1$ indicates that the parity-check equation corresponding to row i is unsatisfied. In the Tanner graph, this means that check node u_i is unsatisfied. A simple decoding strategy is then to identify variable nodes connected to many unsatisfied check nodes and flip the corresponding bits of $\mathbf{r}$. This process can be repeated until the syndrome becomes zero, or until a maximum number of iterations is reached.

LDPC and MDPC Codes

The efficiency of decoding algorithms based on Tanner graphs is closely related to the number of edges in the graph. Since each nonzero entry of H corresponds to an edge in the Tanner graph, sparse parity-check matrices generally lead to more efficient decoding procedures.

This motivates the use of *low-density parity-check* codes, or LDPC codes. In an LDPC code, the parity-check matrix has very small row and column weights. In particular, the row weight is often constant or very small relative to the block length.

We now introduce the notation used for LDPC (and later MDPC) codes.

Definition ((n, r, w)-LDPC Code)**.** *An (n, r, w)-LDPC code is a binary linear code of length n with a parity-check matrix*

$$H \in \mathbb{F}_2^{r \times n}$$

such that every row of H has Hamming weight w. Here n is the code length, r is the codimension or redundancy, so that H has r rows and n columns, w is the row weight of H. If H has full rank, then the dimension of the code is $k = n - r$

Decoding Performance and Attacks

LDPC codes have very efficient decoding algorithms because their Tanner graphs have relatively few edges. However, this low-density structure can be problematic in code-based cryptography.

Recall that a parity-check matrix H for a code C has rows in the dual code $C^\perp$. Therefore, if H has very low-weight rows, then the dual code $C^\perp$ contains very low-weight codewords. If an attacker can find enough low-weight codewords in the dual code, then they may be able to reconstruct a sparse parity-check matrix for the hidden code.

MDPC codes address this issue by increasing the row weight of the parity-check matrix. Instead of using a very small row weight as in LDPC codes, MDPC codes use a moderate row weight such as $w = O(\sqrt{n \log n})$

This makes the hidden parity-check rows harder to recover using low-weight codeword-finding attacks, while still preserving enough sparsity to allow efficient iterative decoding. Thus, MDPC codes trade some decoding performance for improved cryptographic security.

Key Sizes and Quasi-Cyclic Structure

Although MDPC codes help address the low-weight dual-codeword issue, unstructured MDPC codes still have a key-size problem. In code-based cryptography, the public key is often derived from a generator matrix or parity-check matrix. If the matrix is unstructured, then one may need to store or transmit a large matrix.

For an unstructured code, a generator matrix requires storing roughly kn binary entries. This can lead to very large public keys. In the case of unstructured MDPC codes, these keys can be significantly (40-80 times) larger than those obtained from classical Goppa-code-based McEliece systems.

To reduce key sizes, one can use quasi-cyclic structure introduced before. If a matrix is built from circulant blocks, then each circulant block is determined entirely by its first row. Therefore, we only need to store one vector of length r. Equivalently, we can store one polynomial in

$$R = \mathbb{F}_2[x]/\langle x^r - 1\rangle.$$

This reduces storage by roughly a factor of r per circulant block.

QC-MDPC Codes

We now combine the two ideas above: the moderate-density parity-check condition and quasi-cyclic structure. For simplicity, lets consider the case $n = 3r$. Then the code has length $n = 3r$, and we construct a parity-check matrix using three $r \times r$ circulant blocks:

$$H = [H_0 \mid H_1 \mid H_2]$$

where

$$H_i \in \mathbb{F}_2^{r\times r}$$

Assuming H has full rank, the dimension of the code is $k = n - r = 3r - r = 2r$. Instead of describing the circulant blocks directly as matrices, we represent them as polynomials in the ring

$$R = \mathbb{F}_2[x]/\langle x^r - 1\rangle.$$

We sample three polynomials

$$h_0(x), h_1(x), h_2(x) \in R$$

using random fixed-weight sampling. If the desired MDPC row weight is w, then in this simple example we may choose

$$\mathrm{wt}(h_0) = \mathrm{wt}(h_1) = \mathrm{wt}(h_2) = \frac{w}{3}.$$

Therefore, the parity-check matrix can be written as

$$H = [h_0(x) \mid h_1(x) \mid h_2(x)]$$

To construct a systematic generator matrix, we assume that the rightmost block H_2 is invertible (H_2 being nonsingular). Equivalently, in polynomial form, we assume that $h_2(x)$ is invertible in the ring.

The systematic generator matrix is

$$G = \begin{pmatrix} I_r & 0 & (H_2^{-1}H_0)^T \\ 0 & I_r & (H_2^{-1}H_1)^T \end{pmatrix},$$

We can verify that this is indeed a generator matrix by checking that

$$HG^T = 0.$$

Moderate Density Parity-Check McEliece variants

The Moderate Density Parity-Check (MDPC) McEliece cryptosystem is a variant of the classic McEliece. It relies on a secret parity check matrix that is sparse enough to allow efficient error correction via Gallager's bit-flipping algorithm, but whose public equivalent appears random and dense.

System Description

Let n be the code length, r the codimension, w the parity-check row weight scaling at $\mathcal{O}(\sqrt{n \log n})$, and t the error correction ability.

- **Key-Generation.**
    1. The private key: Generate a parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ of an (n, r, w)-MDPC or (n, r, w)-QC-MDPC code with error correction ability t.
    2. The public key: Generate the corresponding generator matrix $G \in \mathbb{F}_2^{(n-r) \times n}$ in systematic form.
- **Encryption.** For a plaintext $m \in \mathbb{F}_2^{(n-r)}$:
    1. Generate a random error vector $e \in \mathbb{F}_2^n$ with $\mathtt{wt}(e) \leq t$.
    2. The ciphertext will be $x \leftarrow mG + e$

- **Decryption.** Let Ψ_H be a t-error correcting decoding algorithm that knows the private key H. For a ciphertext $x \in \mathbb{F}_2^n$:

    1. Get rid of the error by $mG \leftarrow \Psi_H(x)$

    2. Extract the plaintext m from the first $(n-r)$ positions of mG because G was in systematic form and the first $(n-r)$ sized block of it is the identity matrix.

Using a CCA2-secure conversion will prevent security flaws that would otherwise arise from G being in systematic form. Also, this description will not need the public key G to be multiplied by the scrambling matrix S, nor the permutation matrix P.

Security Reductions

Let $\mathcal{F}_{n,r,w}$ be a t-error correcting code family which is either (n, r, w)-MDPC or (n, r, w)-QC-MDPC, $\mathcal{K}_{n,r,w}$ be the key space of $\mathcal{F}_{n,r,w}$, and $\mathcal{H}_{n,r} \supset \mathcal{K}_{n,r,w}$ be the apparent key space of $\mathcal{F}_{n,r,w}$. In the MDPC case, $\mathcal{H}_{n,r}$ would be the set of all full rank matrices in $\mathbb{F}_2^{r\times n}$, and the same in the QC-MDPC case, just restricted to block circulant matrices. Also, let $\mathcal{S}_n(0,t)$ denote the sphere centered in zero of radius t in the Hamming space $\mathbb{F}_2^n$, representing all possible error vectors.

For the reductions, we consider the Niederreiter cryptosystem. This scheme is equivalent in terms of security to the McEliece cryptosystem. There is a generic security reduction for Niederreiter, which we will not go over, but it would be easy to see that that security reduction also holds for the McEliece scheme, at the price of more involved probability space and statements. The following proposition supports that security reduction.

Proposition. *Given the security parameters (n, r, w) and t, if there exists a (T, ϵ)-adversary against $\mathcal{K}_{n,r,w}$ Niederreiter, then there exists either a $(T, \epsilon/2)$-decoder for $(\mathcal{H}_{n,r}, t)$ or a $(T + O(n^2), \epsilon/2)$-distinguisher for $\mathcal{K}_{n,r,w}$ vs. $\mathcal{H}_{n,r}$.*

This proposition dictates that breaking the scheme requires the attacker to solve at least one of the following two problems:

Problem 1. (Code Distinguishing Problem) For parameters $\mathcal{K}_{n,r,w}, \mathcal{H}_{n,r}$ and given a matrix $H \in \mathcal{H}_{n,r}$, is $H \in \mathcal{K}_{n,r,w}$?

Problem 2. (Computational Syndrome Decoding Problem) For parameters $\mathcal{H}_{n,r}$, an integer $t > 0$, and given a matrix $H \in \mathcal{H}_{n,r}$ and a vector $s \in \mathbb{F}_2^r$, find a vector $e \in \mathcal{S}_n(0,t)$ such that $eH^T = s$.

It is enough to assume that none of those problems can be solved efficiently to ensure that no efficient adversary against the main scheme exists. As mentioned in Week 2 notes, Problem 2 is referred to as NP-hard. Therefore, if an attacker attempts to decode the message directly, they face a hard problem. The other remaining threat to the system is the Distinguishing Attack (Problem 1).

Since the public matrices for the MDPC/QC-MDPC McEliece variants do not have an exploitable structure, to distinguish them from random matrices for random linear codes, one needs to determine if there exist low-weight codewords in their dual code.

Problem 3. (Codeword Existence Problem) For parameters $\mathcal{H}_{n,r}$, an integer $w > 0$, and given a matrix $H \in \mathcal{H}_{n,r}$, is there a codeword of weight at most w in the code of generator matrix H?

Ideally, we would like to replace Problem 1 by Problem 3 in the first Proposition. However, this is not directly possible. Nevertheless, this is obtained if we assume the following:

Assumption 1. Solving Problem 1 for parameters $(\mathcal{H}_{n,r}, \mathcal{K}_{n,r,w})$ is not easier than solving Problem 3 for the parameters $(\mathcal{H}_{n,r}, w)$.

Within this assumption, the reduction can be modified to a claim that the $\mathcal{K}_{n,r,w}$-McEliece scheme is at least as hard as either Problem 2 or Problem 3. But only knowing the existence of low-weight codewords is not enough to decode. If one can find those codewords, they can reconstruct the private key and then decode. Thus, consider the computational problem associated with Problem 3:

Problem 4. (Codeword Finding Problem) For parameters $\mathcal{H}_{n,r}$, an integer $w > 0$, and given a matrix $H \in \mathcal{H}_{n,r}$, find a codeword of weight at most w in the code of generator-matrix H.

We will give intuition as to why this problem is polynomially equivalent to Problems 2 and 3.

Lemma. *Problem 3 is polynomially equivalent to Problem 4.*

Intuition. If we have a solver for Codeword Finding (Problem 4), it trivially solves Codeword Existence (Problem 3). If we have a solver for Codeword Existence, we can isolate the actual codeword by iteratively puncturing the code, i.e., forcing specific coordinate dimensions to zero. At each step, we query the solver for Codeword Existence. If it confirms the codeword still exists, we know the punctured coordinate was zero, effectively trapping the exact support and location of the target codeword.

Lemma. *Problem 4 is polynomially equivalent to Problem 2.*

Intuition. Given a Syndrome Decoder, we can leverage it to look for a hidden codeword. We isolate a basis vector of the target code and compute its syndrome relative to the subcode formed by the remaining basis vectors. Feeding this constructed syndrome to the Decoder forces it to locate the missing piece of the codeword, directly yielding the complete target codeword of weight w. As for the other direction, given a Codeword Finder, we can append the target syndrome s as an additional column to the parity-check matrix H. We then task the Finder to locate a slightly larger codeword of weight $w + 1$ in this augmented matrix. To form a valid mathematical codeword that maps to zero, the Finder must select the newly appended column, forcing a 1 in the final coordinate to act as a cancellation mechanism for s. This leaves the remaining w elements of the output as the exact, necessary error vector that decodes the original syndrome.

Proposition. *Given Assumption 1:*

- *Breaking the MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random code.*
- *Breaking the QC-MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random quasi-cyclic linear code.*

This follows directly from the first Proposition and the polynomial equivalence of problems 3-4 and 4-2.

Week 4

Information Set Decoding

Talk by Sam and Roman.

Lee–Brickel (1988)

- Weaker bet: Assume ν errors in information set.
- Enumerate all error vectors with Hamming weight ν in $\mathbb{F}_q^k$, try to decode .

Analysis:

- Per iteration: $O(n^3)$ (Gaussian elimination) + $O(\binom{k}{\nu})$ (# vectors to try) $\cdot O(n-k)$ (checking each candidate if it yields the correct syndrome)
- Success probability per iteration: $\binom{k}{\nu}\binom{n-k}{t-\nu} / \binom{n}{t}$
- Outperforms Prange for small values of ν.

Leon (1988)

- Cost-per-check in Lee–Brickell: Check needs to compute the full syndrome. Naive: $O(k(n-k))$, Optimized: $O(n-k)$ (“Gray code enumeration”)
- Leon’s algorithm brings the cost-per-check down (at the price of a higher number of iterations) by introducing a “zero-window”.
- The zero-window is a zone of $\ell \ll k$ outside the information-set where we assume no errors to occur.
- Pre-filter candidates: Check if they yield a syndrome with 0 errors in the zero-window (Cost $O(\ell)$)
- Full check is only performed on candidates that pass the pre-filter. Cost: $O(\nu(n-k))$,
- Per-iteration-cost: $O(n^3)$ (Gaussian elimination) + $O(\binom{k}{\nu})$ (# vectors to try) $\cdot O(\ell)$ (pre-filter cost) $+ O(\binom{k}{\nu})2^{-\ell}$ (expected # full checks) $\cdot O(\nu(n-k))$ (full-check cost)

- Success probability per iteration: $\binom{k}{\nu}\binom{n-k-\ell}{t-\nu}/\binom{n}{t}$

Dumer (1991)

To find $\mathbf{e}$ of Hamming weight t with $H\mathbf{e}^\top = \mathbf{s}$ (syndrome decoding problem), we can proceed as follows:

- Write $H = (H_1\ H_2)$ with $H_1, H_2 \in \mathbb{F}_q^{(n-k)\times n/2}$. We hope that $t/2$ of the errors of our codeword are in positions belonging to H_1 and $t/2$ belong to positions in H_2
- Find $\mathbf{e}_1, \mathbf{e}_2$ of Hamming weight $t/2$ with $H_1\mathbf{e}_1^\top + H_2\mathbf{e}_2^\top = \mathbf{s}$ as follows
  - Enumerate all error vectors $\mathbf{e}_1$ of Hamming weight $t/2$ and store $(\mathbf{e}_1, H_1\mathbf{e}_1^\top)$ in a list L_1.
  - Enumerate all error vectors $\mathbf{e}_2$ of Hamming weight $t/2$ and store $(\mathbf{e}_2, \mathbf{s} - H_2\mathbf{e}_2^\top)$ in a list L_2.
  - Find a "collision" $(\mathbf{e}_1, \mathbf{x}) \in L_1, (\mathbf{e}_2, \mathbf{x}) \in L_2$. If such a collision has been found, output $(\mathbf{e}_1\ \mathbf{e}_2)$ as solution. This can be implemented for example with hash tables.

Analysis: Runtime for finding one solution:

$$|L_1| = |L_2| = \binom{n/2}{t/2}(q-1)^{t/2}$$

Runtime for finding all solutions (needed for ISD):

$$L_1 \bowtie L_2 := \{(\mathbf{e}_1, \mathbf{e}_2) \mid \exists \mathbf{x}(\mathbf{e}_1, \mathbf{x}) \in L_1, (\mathbf{e}_2, \mathbf{x}) \in L_2\}$$

Runtime: $|L_1| + |L_1 \bowtie L_2|$

$$|L_1 \bowtie L_2| \leq \#\text{solutions to the SDP} \approx \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}.$$

Stern (1989)

Key idea: Take Leon's algorithm, replace iterating weight ν error vectors + pre-filtering with Dumer's algorithm.

- Split IS into two windows with $\nu/2$ errors each.
- Iterate errors $\mathbf{e}_1, \mathbf{e}_2$ of hamming weight $\nu/2$.
- Build lists L_1, L_2 that store $\mathbf{e}_1$ resp. $\mathbf{e}_2$ with the corresponding zero-window component
- Search for all $\mathbf{e}_1, \mathbf{e}_2$ where the zero-window component collides.
- Then $\mathbf{e} := (\mathbf{e}_1\ \mathbf{e}_2)$ is a candidate vector that passes Leon's pre-filter. Perform full check on $\mathbf{e}$

Week 5

Decoding for quasi-cyclic codes

Talk by Camryn.

DOOM Attack

In week 4, we looked at algorithms to solve the syndrome decoding problem; that is, we looked at algorithms which, given a binary parity check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$, a syndrome $s \in \mathbb{F}_2^{n-k}$, and a target weight w, are able to find an error vector $e \in \mathbb{F}_2^n$ such that $He^T = s$ and $w_H(e) = w$. This week, we'll start by looking at a more general instance of that problem: the **D**ecoding **O**ne **O**f **M**any problem.

Decoding One Of Many (DOOM):
Given $H \in \mathbb{F}_2^{(n-k)\times n}$, a set of syndromes $s_1, \ldots, s_M \in \mathbb{F}_2^{n-k}$, and weight w, find an error vector $e \in \mathbb{F}_2^n$ and an index $1 \leq i \leq M$ such that $He^T = s_i$ and $w_H(e) = w$.

Generic DOOM Attack:
For every choice of information set:

1. Run an existing ISD algorithm (e.g., Stern)
2. Check whether the resulting candidate decodes *any* syndrome in $\{s_1, \ldots, s_M\}$.

With every iteration of the algorithm, we have M chances of success, so the success probability of the algorithm total is roughly

$$M \cdot \Pr[\text{ISD algorithm is successful}].$$

This generic attack against the DOOM problem can be easily converted into a (regular) syndrome decoding attack against quasi-cyclic codes. Let H be the parity-check matrix for a quasi-cyclic code of index ℓ, and let s be a syndrome and e be an error vector such that

$$He^T = s.$$

By week 2, we know that

$$H(\ell\text{-shift}(e))^T = \ell\text{-shift}(s).$$

Hence, if the attacker finds a solution for any target s' obtained by shifting s by some multiple of ℓ positions, then they can use this to obtain a solution for the target s. So, the syndrome decoding problem for quasi-cyclic codes reduces to the DOOM problem for generic codes.

Squaring Attack

This is a key-recovery attack on McEliece (when McEliece uses quasi-cyclic codes). For reference, we include the key generation algorithm for McEliece below (encryption and decryption are irrelevant here):

KeyGen():

1. Choose $r \times r$ circulant matrices H_0 and H_1, where H_i has weight w_i in each row. Set $H = [H_0 H_1]$ and set $w = x_0 + w_1$.

2. Compute a generator matrix G satisfying $GH^T = 0$. Specifically, compute

$$P = \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{\ell-2} \end{bmatrix} = \begin{bmatrix} (H_{\ell-1}^{-1} H_0)^T \\ (H_{\ell-1}^{-1} H_1)^T \\ \vdots \\ (H_{\ell-1}^{-1} H_{\ell-2})^T \end{bmatrix}$$

   and set $G = [IP]$, where I is the identity matrix.

3. Set the public key to be G and the secret key to be H.

Note that our parity check matrix P is NOT the same as the permutation in the original McEliece construction. The permutation matrix is not necessary with quasi-cyclic codes. Also, for the purposes of this attack, we must assume that r is even. Finally, denote the code generated by P as C.

Goal of the attack:
Given P, find H.

"Background" knowledge of the attack:

- Since $H = [H_0 H_1]$ for circulant matrices H_0 and H_1, it suffices to recover one row of H (from which H can then be recomputed).

- Any row of $[H_1 H_0]$ is a low-weight codeword of the code generated by G. This is not true in general for quasi-cyclic codes, but is true in this case since H only has two blocks. The proof is very boring and tedious symbol-pushing, so I'll skip it here.

- With high probability, if we find *any* low-weight codeword, it will be a row of $[H_1 H_0]$. To see this, first assume that the weight distribution of the codewords is the same as the weight distribution of $2r$-bit strings. Since we have a total of 2^r codewords, $\binom{2r}{w}$ strings of weight w, and 2^{2r} $2r$-bit strings, then there should be

$$2^r \left(\frac{\binom{2r}{w}}{2^{2r}} \right) = \frac{\binom{2r}{w}}{2^r},$$

  which is very small.

- Since all of our matrices are circulant, we can construct them from their first row only. We can represent this row as a polynomial (and can work over the ring $\frac{\mathbb{F}_2[x]}{x^r+1}$ - proof of isomorphism in week 2). We'll use the following notation for our code $\mapsto$ polynomial mappings:

  - $H_0 \mapsto h_0(x)$
  - $H_1 \mapsto h_1(x)$
  - $P_0 \mapsto p(x)$

- By definition of P_0, we get that
$$H_1 P_0^T = H_0.$$
This means that, under our polynomial mapping,
$$h_1(x)p(x) \equiv h_0(x) \mod (x^r + 1).$$
Squaring both sides gives us that
$$h_1^2(x)p^2(x) \equiv h_0^2(x) \mod (x^r + 1).$$
Since circulant matrices (and polynomials) commute, we get that that $h_1^2(x)h_0^2(x)$ is still a codeword of the code generated by P^2.

- Since we're working over $\mathbb{F}_2$, for any polynomial $f(x) = a_0 + a_1 x + \cdots + a_{r-1}x^{r-1}$, we get that
$$f(x)^2 = a_0^2 + a_1^2 x^2 + \cdots + a_{r-1}^2 x^{2(r-1)} \mod (x^r - 1).$$
Notably, the squaring operation maps the all non-zero coefficients of $f(x)$ (which could occur at any index) onto coefficients of an index that is a multiple of 2, which limits the number of non-zero coefficients in $f^2(x)$. If we repeatedly squared $f(x)$ (say, q times), we would end up with a polynomial for which all of the non-zero coefficients lie on indices that are multiples of 2^q.

Sub-algorithm: Bounded-weight polynomial square roots
Let $u = (a, b) \in C$ such that $aP = b$ and let $u_2 = (a^2, b^2) \in C_2 = C^2$. Further, let $w_H(u) = w_1$ and let $w_H(u_2) = w_2$. For this algorithm, suppose that we know u_2, w_1, and w_2. The goal is to find u.

Notice that the x^i and and $x^{i+\frac{r}{2}}$ terms of u both square to be x^{2i} terms in u_2. So, if the coefficient of the x^{2i} term in u_2 is 1, then the coefficient of exactly one of the x^i and $x^{i+\frac{r}{2}}$ terms in u must have been 1, while the other one must have been 0. On the other hand, if the coefficient of the x^{2i} term in u_2 is 0, then either the coefficients of the x^i and $x^{i+\frac{r}{2}}$ terms in u must both have been 1, or must both have been 0. So, since each non-zero coefficient in u_2 corresponds to exactly one non-zero coefficient in u, we get that $w_2 \leq w_1$.

If $w_2 < w_1$ (which happens with high probability), then $w_1 - w_2$ counts the number of coefficients of 1 that have been "lost" in the squaring attack. From u_2, we can see exactly which coefficients "survive" the squaring. Let S be the set of indices of the "surviving" coefficients (i.e., $S = \text{Supp}(u_2)$). Note that $|S| = w_2$.

We can construct a new code C' by deleting all coordinates from C corresponding to S (that is, if $2i \in S$, then both i and $i + \frac{r}{2}$ are deleted from C). Clearly C' is still a subspace, hence it's a linear code, and since we removed $2w_2$ coordinates, it's a $[2r - 2w_2, r]$-linear code. Further, C' contains the codeword u' (the transformation of u under the deletion map we just defined), which will have weight $w_1 - w_2$. This weight is (probably) quite small, and in this new smaller code, codewords of this weight can be computed very efficiently using a generic ISD algorithm. Similarly to before/later, with very high probability we will obtain the vector we are looking for and not a "false positive".

This gives us the non-zero coefficients of u that did not cancel out with each other during the squaring process. We can find the ones that did cancel out using boring linear algebra, and the system of equations
$$aP = b$$
$$a_{i+\frac{r}{2}} + a_i = 1 \; \forall 0 \leq i \leq \frac{r}{2}.$$

The attack:

1. **Squaring the code**
   Let d be the maximum integer such that 2^d divides r (since r is even, note that $d > 0$). Choose $q \in \{1, \ldots, d\}$, and compute $p^{2^q}(x)$ (i.e., square the code q times). By the discussion above, we know that
   $$h_1^{2^q}(x)p^{2^q}(x) \equiv h_0^{2^q}(x) \mod (x^r + 1),$$
   and that $w_H(h_1^{2^q}(x)) \leq w_H(h_1(x))$ and $w_H(h_0^{2^q}(x)) \leq w_H(h_0(x))$ (and that these weights have probably decreased - in the paper they show a simulated weight distribution).

2. **Finding a low-weight polynomial**
   Use a generic ISD algorithm (e.g., Stern) to find a low-weight polynomial (weight at most $2w_q$ for a chosen parameter w_q) in the code generated by P^{2^q}. With high probability, this will be a cyclic shift of $h_0^{2^q}(x)$; in the paper, the show that the expected number of "false positives" is
   $$2^{-\frac{r}{2^q}+1}\binom{r/2^q}{w_q/2}^2.$$

3. **Reconstructing the polynomials $h_0(x)$ and $h_1(x)$**
   We can use the nice algorithm above to do this. There are two options:

   - Use the nice algorithm above on C^{2^ℓ} rather than on C^2, and directly compute h_0 and h_1.
   - Use the nice algorithm above ℓ times in a row, to undo each squaring operation separately.

   The first option takes fewer steps, but if q is large then the second option tends to be easier.

The complexity of the attack is given by

$$C^*(r,w) \leq \min_{q,m}\left\{\frac{1}{\Psi(r,w,m,q)^2}\left(C_{ISD}(\frac{2r}{2^q},\frac{r}{2^q},w-4m)\right.\right.$$
$$\left.\left.+C_{ISD}(2r-2^q(w-4m),r,4m)\cdot(\Psi(r,w,m,q)^2+\mu^*(r,q,2m))\right)\right\}.$$

where

- $\Psi(r,w,m,q)$ is the probability of m "collisions" (i.e., two coefficients of u that cancel out in u_2) in q squarings of a polynomial of weight w;
- $C_{ISD}(n,k,w)$ is the complexity of the generic ISD algorithm when used to find a codeword of weight w in an $[n,k]$-linear code;
- $\mu*(r,q,w)$ is the expected number of "false positives" of weight w from the "finding low-weight polynomials" step of the algorithm.

Week 6

Hamming Quasi Cyclic (HQC)

Week 7

HQC implementations/optimizations

Talk by Maggie Simmon and Ganyu (Bruce) Xu

BCH-view Reed-Solomon decoder

Polynomial multiplication

References

Bibliography:

[1] C. Aguilar-Melchor, O. Blazy, J. -C. Deneuville, P. Gaborit and G. Zémor. Efficient Encryption From Random Quasi-Cyclic Codes. In *IEEE Transactions on Information Theory*, vol. 64, no. 5, pp. 3927–3943, 2018.

[2] T. Debris-Alazard. *Code-based Cryptography Lecture Notes*, 2023.

[3] J. Dong, Y. Hou, S. Wang, L. Sha, F. Xiao, Z. Dong, and J. Lin. HIGH: Harnessing GPU Parallelism for Optimized HQC Performance. In *IACR Cryptology ePrint Archive*, 2026.

[4] HQC Team. *Hamming Quasi-Cyclic (HQC)*, NIST Submission, 2025.

[5] L. Huguenin-Dumittan and S. Vaudenay. FO-like Combiners and Hybrid Post-Quantum Cryptography. In *Cryptology and Network Security: 20th International Conference*, *CANS 2021*, pp. 225–244, 2021.

[6] C. Löndahl, T. Johansson, M. Koochak Shooshtari, M. Ahmadian-Attari, and M. Reza Aref. Squaring attacks on McEliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Designs, Codes and Cryptography* , vol. 80, pp. 359–377, 2016.

[7] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto. *MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes*. 2013 IEEE International Symposium on Information Theory, 2013.

[8] NIST. *Post-Quantum Cryptography: Additional Digital Signature Schemes*.

[9] R. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[10] N. Sendrier. Decoding One Out of Many. *Post-Quantum Cryptography. PQCrypto 2011. Lecture Notes in Computer Science*, vol. 7071, Springer, 2011.

[11] T. Wang, Q. Teng, A. Wang, J. Zhang, B. Pang, C. Zhao, S. Hu, and X, Wang. HARE: Compact HQC via Distance-Informed Erasure Decoding. In *Cryptology ePrint Archive*, *Paper 2026/544*, 2026.

[12] V. Weger, N. Gassner, and J. Rosenthal. *A Survey on Code-based Cryptography*. arXiv, 2024.