UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Elettronica
Curriculum: Smart and Secure Communication Networks

PROGETTAZIONE DI DECODER EFFICIENTI PER HQC, UN CRITTOSISTEMA POST-QUANTUM BASATO SU CODICI

Designing efficient decoders for HQC, a post-quantum code-based cryptosystem

Relatore:
Prof. Paolo Santini

Correlatore:
Prof. Marco Baldi

Candidato:
Nicholas Lilla
Mat. 1105584

Anno Accademico 2022-2023

Contents

Abstract

Nella tesi, redatta in lingua inglese, si conduce uno studio dettagliato su HQC (Hamming Quasi Cyclic), un crittosistema post-quantum basato su codici. Inizialmente, si offre una panoramica dei concetti fondamentali della teoria dei codici, per poi procedere con l'analisi delle diverse famiglie di codici impiegate in HQC: codici a ripetizione, codici BCH, codici Reed-Solomon e, infine, i codici Reed-Muller.

Successivamente, si esamina con dovizia di particolari il background matematico su cui si fonda HQC, con attenzione alla definizione del problema SDP (Syndrome Decoding Problem) nella sua versione decisionale. Inoltre, vengono presentate le varianti di questo problema, ovvero 2-QCSD e 3-QCSD con parità e cancellazioni. Nel prosieguo, senza pretesa di esaustività, si affronta il tema della teoria della complessità. In particolare, vengono definite diverse classi di complessità (P, NP, NP-complete e NP-Hard) ed i relativi problemi noti, tra cui la fattorizzazione di interi, SDP, SAT, vertexCover, setCover e Halting Problem. Inoltre, si assume ragionevolmente che le varianti di SDP considerate appartengano alla classe di complessità di SDP, cioè la classe dei problemi NP-Complete.

Il lavoro prosegue con una minuziosa descrizione delle basi teoriche di HQC, delineando chiaramente la distinzione tra la versione PKE (Public Key Encryption) e KEM (Key Encapsulation Mechanism). Vengono forniti dettagli approfonditi riguardanti i parametri e le diverse configurazioni dei quattro round della competizione NIST, comprendenti le dimensioni di chiave pubblica, chiave segreta, testo cifrato, e l'eventuale chiave segreta condivisa. In aggiunta, si offre un approfondito e rigoroso studio matematico del DFR (Decoding Failure Rate) relativo al decoder HQC originale.

Il culmine del lavoro di tesi è rappresentato dalla presentazione del nuovo decoder HQC, il quale incorpora il tradizionale decoder HQC arricchendolo con un pre-filtraggio del rumore basato su una strategia di correlazione. È cruciale sottolineare che la concezione di questo nuovo decoder HQC è il cuore pulsante del lavoro, la sua parte più significativa e il punto focale dell'intera ricerca. Tale decoder nasce da una considerazione strettamente ancorata alla teoria dell'informazione: c'è informazione che nella decodifica classica di HQC rimane inutilizzata, quindi, il suo impiego si traduce inevitabilmente in un notevole aumento dell'efficienza. La tesi offre, pertanto, un'analisi matematica dettagliata del DFR per il nuovo decoder, comparandolo esplicitamente con il DFR del decoder HQC tradizionale. In aggiunta, si valuta attentamente la complessità computazionale del nuovo decoder, evidenziando in modo esplicito il suo esiguo incremento rispetto alla controparte originale.

In conclusione, vengono presentate simulazioni implementate in C e Python a conferma dei risultati teorici, con l'intento di evidenziare i benefici derivanti dall'introduzione del decoder proposto. In particolare, il nuovo decoder consente di ridurre le dimensioni della chiave pubblica e del testo cifrato, contribuendo così a ridurre la quantità di dati trasmessi su canale pubblico. Va notato che l'entità di questa riduzione è fortemente influenzata dalla configurazione utilizzata, pertanto, nei prossimi capitoli di tale tesi, questo aspetto verrà esaminato dettagliatamente per le diverse configurazioni di HQC.

Infine, nella parte finale della tesi sono delineati alcuni scenari di interesse pratico in cui l'introduzione del nuovo decoder può apportare benefici, specialmente in contesti in cui si adoperano chiavi effimere per garantire la Perfect Forward Secrecy (PFS), come ad esempio nel protocollo TLS (Transport Layer Security) e nelle VPN (Virtual Private Network).

Abstract

This thesis explores the Hamming Quasi Cyclic (HQC) post-quantum code-based cryptosystem, delving into coding theory principles and examining code families such as repetition, BCH, Reed-Solomon, and Reed-Muller codes. Additionally, mathematical aspects, particularly the Syndrome Decoding Problem (SDP) and its variants (2-QCSD, 3-QCSD), are scrutinized, with a brief introduction of complexity theory.

The theoretical foundations of HQC, distinguishing between Public Key Encryption (PKE) and Key Encapsulation Mechanism (KEM) versions, are thoroughly detailed. The parameters and configurations of NIST competition rounds are extensively discussed, accompanied by a mathematical analysis of the Decoding Failure Rate (DFR) for the original HQC decoder.

The focal point of the thesis is the groundbreaking introduction of a new HQC decoder. This innovative decoder incorporates noise pre-filtering based on a correlation strategy, harnessing information completely overlooked in the classic HQC decoder. This pivotal addition unequivocally ensures greater efficiency. Furthermore, the DFR for the new decoder is meticulously analyzed and compared with the classical HQC decoder.

Simulations, conducted using both C and Python code, validate theoretical findings, highlighting the proposed decoder's advantages in reducing public key and ciphertext sizes leading to a decrease in data transmitted over a public channel. The degree of reduction is configuration-dependent, prompting a detailed examination of various HQC configurations in subsequent chapters.

Finally, some practical scenarios are outlined in which the introduction of the new decoder can bring benefits, especially in contexts where ephemeral keys are used to ensure Perfect Forward Secrecy (PFS), such as in the TLS (Transport Layer Security) protocol and in Virtual Private Networks (VPNs).

Introduction

In recent years, significant strides have been made in the field of quantum computing, which employs the intricate principles of quantum mechanics to tackle complex mathematical problems. A quantum computer is a machine that employs quantum-physical phenomena to perform computations in a way that is fundamentally different from a "normal" classical computer. While a classical computer is, at any point in time, in a fixed state, such as a bit string representing its memory contents, the state of a quantum computer can be a "mixture", a so-called superposition, of several states [1]. The potential power of quantum computers far surpasses that of classical computers. Consequently, if large-scale quantum computers are ever built, they will have the capability to undermine many of the cryptosystems that currently safeguard our digital communications. Some hard mathematical problems which can be solved in polynomial time with quantum computers are at the basis of many widespread cryptographic primitives and protocols, like Rivest, Shamir, Adleman (RSA), ElGamal, Digital Signature Algorithm (DSA), Elliptic Curve Digital Signature Algorithm (ECDSA), Diffie-Hellman and others [2] [3].

In the field of quantum algorithm development, two pioneering algorithms have formed a solid foundation for potentially breaking today's theoretically grounded public-key cryptosystems. In 1994, Shor introduced an efficient polynomial-time algorithm designed for solving integer factorization and discrete logarithm problems, based on the existence of quantum computers [4]. In 1996, Grover introduced a quantum algorithm characterized by a square

root of N complexity for searching an element within an unsorted database comprising N records [5]. Upon realization on quantum computers, Grover's algorithm has the potential to undermine symmetric-key cryptosystems. To counteract attacks based on Grover's algorithm, it becomes necessary to double the key sizes in order to maintain a comparable level of security against classical computers.

In line with the preceding statements, the implications of a quantum computing breakthrough, which experts believe is merely a matter of time, are profound, jeopardizing the confidentiality and integrity of modern communications. While quantum computers have yet to materialize in the present, it remains imperative to diligently pursue the establishment of cryptographic systems that can withstand quantum computer menace, and by extension, traditional computing threats. These cryptographic solutions should seamlessly integrate with and operate within existing networks and communication systems. As a response to this impending challenge, the field of post-quantum cryptography (also called quantum-resistant cryptography) has emerged.

Recognizing the urgency of preparing for the post-quantum era, the National Institute of Standards and Technology (NIST), a pivotal agency within the United States government responsible for technology management, initiated in 2013 a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms [6]. Over the years, this cryptographic competition has progressed through a series of rigorous rounds, each designed to scrutinize various aspects of cryptography, including public-key encryption (PKE), key encapsulation mechanisms (KEMs), and digital signature schemes.

In a bid to ensure an exhaustive selection process, this contest was meticulously divided into three distinct rounds for each of the aforementioned categories. As of the present moment, NIST has successfully concluded the third round of the Post-Quantum Cryptography (PQC) standardization process.

PQC Algorithm	Status	Type	PKE/KEM vs. Signature
CRYSTALS-Kyber	Standard	Lattice	PKE/KEM
CRYSTALS-Dilithium	Standard	Lattice	Signature
FALCON	Standard	Lattice	Signature
SPHINCS	Standard	Hash	Signature
HQC	Round 4	Code	PKE/KEM
BIKE	Round 4	Code	PKE/KEM
Classic McEliece	Round 4	Code	PKE/KEM
SIKE	Broken	Isogeny	PKE/KEM

Table 1: Current state of the NIST PQC Standardization Process

Four candidate algorithms (CRYSTALS-KYBER as PKE/KEM, CRYSTALS-Dilithium, FALCON and SPHINCS+ as digital signature schemes) have been selected for standardization [7]. Additionally, four other algorithms (BIKE, Classic McEliece, HQC, and SIKE) advanced into a further round, the fourth, to be chosen for Key Encapsulation Mechanisms (KEMs). To summarize, refer to Table 1 [8]. It's worth noting that the fourth round is currently underway, and SIKE has been subjected to an attack, compromising its security. This phase is crucial in order to identify a robust Key Encapsulation Mechanism, with a keen focus on exploring alternatives to the prevailing lattice-based schemes that have held prominence up to the conclusion of the third round.

Notably, within the realm of these alternatives, code-based cryptography, underpinned by principles drawn from coding theory, has emerged as a standout contender. BIKE and HQC, both rooted in structured codes, present compelling options for a KEM that does not rely on lattices. It is anticipated that, upon the culmination of this fourth round, NIST will select at most one of these two candidates for formal standardization.

Our contribution

The focal point of this thesis work is the design of a new decoder for HQC. In particular, our contribution lies in defining a novel decoding algorithm based on correlation. Specifically, we observed that in the decoding algorithm proposed in HQC, the knowledge of the secret key was entirely disregarded, even though it is known to the legitimate user intending to decrypt. In HQC, the overall error pattern added to the codeword is assumed to be entirely random during the decryption phase. However, as previously mentioned, we observed that this is not the case in reality. For this reason, the fundamental idea is to leverage this information that is completely overlooked in HQC to enhance the efficiency of the decoding algorithm. To this end, what we have introduced is a new decoding mechanism that involves the introduction of a preliminary noise filtering stage with a correlation-based strategy. We will delve into the details of this new algorithm extensively throughout this work. However, intuitively, it is reasonable to expect that utilizing previously overlooked information may contribute to achieving higher performance.

In this context, better performance inevitably means that, with an equal Decoding Failure Rate (DFR), the size of the ciphertext and the public key can be reduced. As an immediate application, we would like to emphasize that the proposed decoder can replace the traditional HQC decoder in all those numerous practical contexts where ephemeral keys are required, such as in Transport Layer Security (TLS) or Virtual Private Networks (VPNs).

To conclude, it is worth noting that our theoretical curve analysis of the DFR enhances comprehension of the decoder's performance. However, establishing an upper bound for the DFR is crucial for proving the IND-CCA (and IND-CCA2) property. A formal proof of IND-CCA necessitates a demonstrable upper bound on success probability; lacking this impedes the claim of IND-CCA.

Thesis organization

The organization of this thesis is structured to provide a clear and logical progression of the topics discussed. In Chapter 1, we lay the groundwork by introducing fundamental concepts and terms that will be essential for comprehending the rest of the thesis. Chapter 2 delves deeper into the world of Coding Theory. Specifically, we focus on a family of codes known as linear block codes, which includes cyclic codes and related variations like shortened cyclic codes and quasi-cyclic codes. On top of that, we also offer a brief introduction to non-binary linear codes for context. Chapter 3 shifts the focus to the codes employed in HQC (Hamming Quasi Cyclic), a post-quantum code-based cryptosystem. This chapter delves into the inner mechanics of HQC codes, dissecting their design and exploring their encoding and decoding mechanisms, affording an in-depth understanding of the cryptosystem's core operations. In chapter 4 we turn our attention to important security concepts and we also take a closer look at the mathematical problems upon which HQC is constructed. In Chapter 5 we provide an in-depth exploration of how HQC is designed, highlighting the various settings and parameters used in different NIST rounds. In Chapter 6 we offer a detailed exploration of the core idea that motivated this thesis, the proposal of a new decoder for HQC based on correlation. We also assess the complexity of the introduced decoder and discuss its Decoding Failure Rate (DFR). In Chapter 7, we present the results of empirical experiments conducted to assess the performance of the newly introduced decoder. These findings offer a valuable empirical assessment of the decoder's real-world applicability and effectiveness, which are further described in the subsequent chapter, Chapter 8. In the last chapter, Chapter 9, we wrap up our study. Here, we bring together the most important things we've discovered in this thesis. We also suggest some ideas for future research, encouraging more exploration and innovation in the field of post-quantum cryptography.

Chapter 1

Notation and preliminaries

Polynomials Throughout the thesis, given a positive integer n, we denote by $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n - 1)$ the polynomial ring consisting of polynomials of maximum degree $n-1$ with coefficients in $\mathbb{Z}_q = \{0, 1, \cdots q-1\}$. As usual, for q being a prime number, $\mathbb{F}_q = \mathbb{Z}_q$ denotes the finite field of order q. Additionally, we denote by $\mathcal{R}_q(w) = \{\mathbf{a} \in \mathcal{R}_q \mid \mathrm{wt}(\mathbf{a}) = w\}$, where $\mathrm{wt}(\cdot)$ denotes Hamming weight, the set of polynomials in $\mathcal{R}_q$ with weight w. When $q = 2$, we ease notation and simply indicate the polynomial ring as $\mathcal{R}$. Sometimes, we will view the elements of $\mathcal{R}_q$ as vectors over $\mathbb{Z}_q$, relying on the following canonical representation:

$$\sum_{i=0}^{n-1} a_i x^i = \mathbf{a}(x) \in \mathcal{R}_q \iff (a_0, a_1, \cdots, a_{n-1}) = \mathbf{a} \in \mathbb{Z}_q^n.$$

Using the canonical representation mentioned above, occasionally, we denote the set of binary n-tuples, namely $\mathbb{F}_2^n$, as $V = \mathcal{R}$. The support of a polynomial $\mathbf{a}$, that is, the set with the indexes of set coefficients, is indicated as $\mathrm{supp}(\mathbf{a})$. Sporadically, given two polynomials $\mathbf{a}$ and $\mathbf{b}$ in $\mathcal{R}_q$, we indicate the j-th coefficient of their product $\mathbf{c} \in \mathcal{R}_q$ as $(\mathbf{ab})_j$. Formally:

$$c_j = (\mathbf{ab})_j = \sum_{i+k \equiv j \bmod n} a_i \cdot b_k, \quad \text{for } j \in \{0, 1, \cdots n-1\}$$

Circulant Matrix Let $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1}) \in \mathbb{F}_2^n$, then the circulant matrix induced by $\mathbf{v}$ is defined and denoted as follows:

$$\mathbf{rot}(\mathbf{v}) = \begin{pmatrix} v_0 & v_{n-1} & \dots & v_1 \\ v_1 & v_0 & \dots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} & v_{n-2} & \dots & v_0 \end{pmatrix} \in \mathbb{F}_2^{n \times n}$$

Each column is the cyclically shifted version of a downward position of the column to its left. Similarly, each row is the cyclically shifted version of a rightward position of the row above. Since a circulant matrix is uniquely determined by its first column (or row), if its coefficients are in $\mathbb{Z}_q$, it is possible to establish a ring isomorphism between the circulant matrix of size n and $\mathcal{R}_q$. Specifically, we can identify the elements of a circulant matrix with first column $(v_0, v_1, \cdots, v_{n-1})$ with a corresponding element of $\mathcal{R}_q$ through the following bijective map:

$$\Phi : \begin{pmatrix} v_0 & v_{n-1} & \dots & v_1 \\ v_1 & v_0 & \dots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} & v_{n-2} & \dots & v_0 \end{pmatrix} \rightarrow v_0 + v_1 \cdot x + \cdots v_{n-1} \cdot x^{n-1} \in \mathcal{R}_q$$

For $q = 2$, it turns out that there exists an isomorphism between $\mathcal{R}$ and the set of circulant matrices with size n and elements in the binary finite field $\mathbb{F}_2$. The null element (i.e., the all zero vector) and the identity (i.e., the vector $(1, 0, \cdots, 0)$) for $\mathcal{R}$ will be indicated as $\mathbf{0}$ and $\mathbf{1}$, respectively. Furthermore, in a circulant matrix of size n, enumerating rows and columns from 0, the i-th row, when read from left to right, is identical to the $(n - i)$-th column when read from bottom to top. This observation leads to an intriguing property of circulant matrices: all rows and all columns have the same Hamming weight, meaning they contain the same number of 1s.

Probability Distributions Given some set A, we write $a \xleftarrow{\$} A$ when a is drawn uniformly at random from the elements of A. We use $\mathscr{B}_{n,\rho}$ to indicate the Bernoulli distribution over $\mathbb{F}_2^n$ with parameter ρ, i.e., the distribution that returns vectors of length n and such that any entry is 1 with probability ρ and 0 with probability $1-\rho$. If $\mathbf{a} \in \mathbb{F}_2^n$ (or, equivalently, $\mathbf{a} \in \mathcal{R}$) is distributed according to $\mathscr{B}_{n,\rho}$, we write $\mathbf{a} \sim \mathscr{B}_{n,\rho}$. The probability that $\mathbf{a}$ has Hamming weight w corresponds to

$$f_{n,\rho}(w) = \binom{n}{w} \rho^w (1-\rho)^{n-w}.$$

Security Level and Big-O Notation We say that a cryptographic algorithm achieves a Security Level (SL) of x bits if the most efficient attack has a computational cost of 2^x. Throughout the thesis, we employ big-O notation to express the computational cost of an algorithm. Specifically, an algorithm's computational cost is defined as the number of operations required for its execution. This number of operations is expressed as a function of the input size. Let n denote the length of an algorithm's input. A function $f(n)$ is considered to be a big-O of $g(n)$ if the limit as n tends to infinity of $f(n)$ over $g(n)$ is less than a positive constant α. Formally:

$$f(n) \in O(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} < \alpha.$$

When $f(n) = O(g(n))$ we say that $g(n)$ is an asymptotic upper bound for $f(n)$, to emphasize we are suppressing constant factors.

Let $\mathcal{A}$ be an algorithm with input size equal to n, and let $T(n)$ be the function describing the computational cost of the algorithm as the input size varies. If $T(n) = O(n^\alpha)$ for a constant α, we say that $\mathcal{A}$ has polynomial time complexity. Instead, if $T(n) = O(2^{n\cdot\beta})$ for a constant β, we say that $\mathcal{A}$ has exponential time complexity.

Chapter 2

Rudiments of Coding Theory

We will now proceed to explore some fundamental definitions and properties about coding theory, a specialized field within information theory. Specifically, coding theory is dedicated to the study and analysis of error-correcting codes. These codes are essential for ensuring the accurate and reliable transmission of information across various communication channels and data storage systems. Firstly, it is essential to establish a clear distinction between two structurally different types of codes: block codes and convolutional codes. Furthermore, within the realm of block codes, a finer categorization can be made, differentiating between linear and non-linear block codes. Notably, non-linear block codes, while theoretically intriguing, remain rarely employed in practical applications and have been subject to comparatively limited research efforts.

In this chapter, we embark on a comprehensive exploration of binary linear block codes [9], shedding light on their prominent characteristics. It is important to note that our discussion, while in-depth, may not cover every aspects of these codes, but it will certainly provide a strong foundation and understanding of their key aspects. Furthermore, our focus will not be limited solely to binary linear block codes, indeed, in the last section of this chapter we will also extend our attention to non-binary codes.

2.1 Introduction to Linear Block Codes

We assume that the output of an information source is a continuous sequence of symbols over $\mathbb{F}_2$, referred to as the information sequence. In block coding, this sequence gets divided into message blocks, each of uniform length, with every message block containing k information bits. This division results in a total of 2^k unique messages. At the channel encoder, each input message $\mathbf{u} = (u_0, u_1, \cdots u_{k-1})$ is encoded into a longer binary sequence $\mathbf{v} = (v_0, v_1, \cdots v_{n-1})$ of n binary digits with $n > k$. This elongated binary sequence $\mathbf{v}$ is called the codeword of the message $\mathbf{u}$. Since there are 2^k distinct messages, there exist 2^k codewords, with each codeword corresponding to a unique message. This set of 2^k codewords is described as constituting an $[n,k]$ block code. For a block code to serve its intended purpose effectively, it's imperative that the 2^k codewords associated with the 2^k distinct messages maintain their distinctiveness. The additional $n-k$ binary symbols introduced to each input message by the channel encoder are referred to as redundant bits. These serve the primary purpose of endowing the code with the ability to detect and correct transmission errors resulting from channel noise or interference. An essential consideration in designing the channel encoder is how to generate these redundant bits in a manner that enhances the error-correcting capabilities of the code.

The code rate, denoted as R and defined as the ratio $R = \frac{k}{n}$ represents the average number of information bits carried by each code bit. For a block code with length n and 2^k codewords, the encoding and decoding processes can become notably intricate, particularly when k assumes substantial values, unless the code exhibits certain structural characteristics. This complexity is rooted in the encoder's requirement to store all 2^k codewords and the decoder's need for a decoding table containing 2^n entries to estimate the transmitted codeword. As such, it becomes imperative for us to shift our focus toward block codes that can be implemented in a feasible manner, as the practicality

of the system is a critical consideration. One structural attribute that stands out as highly preferable for a block code is linearity.

Definition 2.1 (Binary linear block code)**.** *A binary block code $\mathcal{C}$ of length n with 2^k codewords is called an [n,k] linear block code if its 2^k codewords form a k-dimensional subspace of the vector space $\mathcal{R}$ of all the n-tuple over $\mathbb{F}_2$.*

Due to the linearity property, given two codewords $\mathbf{u}$ and $\mathbf{v} \in \mathcal{C}$, it follows that their sum, $\mathbf{x} = \mathbf{u} + \mathbf{v}$ also belongs to $\mathcal{C}$.

2.1.1 Generator and Parity-Check Matrix

According to Definition 2.1, each codeword $\mathbf{v} \in \mathcal{C}$ can be expressed as a linear combination of k linearly independent codewords in $\mathcal{C}$, namely $\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_{k-1}$, which serve as a basis for the code. Using this basis, encoding can be done as follows. Let $\mathbf{u} = (u_0, u_1, \cdots, u_{k-1})$ be the message to be encoded. The codeword $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ for this message is given by the following linear combination of $\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_{k-1}$, with the k message bits of $\mathbf{u}$ as the coefficients:

$$\mathbf{v} = u_0 \cdot \mathbf{g}_0 + u_1 \cdot \mathbf{g}_1 + \ldots + u_{k-1} \cdot \mathbf{g}_{k-1}.$$

We may arrange the k linearly independent codewords, $\mathbf{g}_0, \mathbf{g}_1, \cdots, \mathbf{g}_{k-1}$ of $\mathcal{C}$ as rows of a $k \times n$ matrix over $\mathbb{F}_2$ as follows:

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{pmatrix} = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix} \tag{2.1}$$

Then, the codeword $\mathbf{v}$ for a message $\mathbf{u}$ can be expressed as the matrix product of $\mathbf{u}$ and $\mathbf{G}$, in symbols, $\mathbf{v} = \mathbf{u} \cdot \mathbf{G}$.

Definition 2.2 (Generator Matrix)**.** *We say that $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ is a Generator Matrix for the [n,k] code $\mathcal{C}$ if $\mathcal{C} = \{\mathbf{m} \cdot \mathbf{G} \mid \mathbf{m} \in \mathbb{F}_2^k\}$*

$\mathcal{C}$ is spanned by the rows of $\mathbf{G}$, therefore, it is called the row space of $\mathbf{G}$. In general, an $[n,k]$ linear code has more than one basis, therefore, a generator matrix of a given $[n,k]$ linear code is not unique. Since a binary $[n,k]$ linear code $\mathcal{C}$ is a k-dimensional subspace of $\mathcal{R}$, its null (or dual) space, denoted $\mathcal{C}_d$, is an $(n-k)$-dimensional subspace of the same vector space given by the following set of n-tuples:

$$\mathcal{C}_d = \{\mathbf{w} \in \mathbb{F}_2 : \langle \mathbf{w}, \mathbf{v} \rangle = 0 \text{ for all } \mathbf{v} \in \mathcal{C}\}$$

where $\langle \mathbf{w}, \mathbf{v} \rangle$ denotes the inner product of $\mathbf{w}$ and $\mathbf{v}$.

The code $\mathcal{C}_d$ can be regarded as a binary $[n,\ n-k]$ linear code, known as the dual code of $\mathcal{C}$. This code consists of $n-k$ linearly independent codewords, namely $\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_{n-k-1}$. Therefore, every codeword in $\mathcal{C}_d$ can be expressed as a linear combination of these $n-k$ linearly independent codewords, which constitute a basis for $\mathcal{C}_d$. As done previously, we may arrange the $n-k$ linearly independent codewords, $\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_{n-k-1}$ of $\mathcal{C}_d$ as rows of a $(n-k) \times n$ matrix over $\mathbb{F}_2$ as follows:

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{pmatrix} = \begin{pmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \dots & h_{n-k-1,n-1} \end{pmatrix} \tag{2.2}$$

Then $\mathbf{H}$ is a generator matrix of the dual code $\mathcal{C}_d$ of the binary $[n,k]$ linear block code $\mathcal{C}$.

Definition 2.3 (Parity-Check Matrix)**.** *Given an [n,k] code $\mathcal{C}$, we say that $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ is a Parity-Check Matrix for $\mathcal{C}$ if $\mathbf{H}$ is a generator matrix of the dual code $\mathcal{C}_d$.*

Since $\mathbf{H}$ is the generator matrix of the dual code, $\mathcal{C}$ is said to be the null space of $\mathbf{H}$. Therefore, a linear code is uniquely specified by two matrices, a

generator matrix and a parity-check matrix. In particular, the two matrices are deeply linked to each other, specifically $\mathbf{G} \cdot \mathbf{H}^{\mathrm{T}} = 0$.

Furthermore, in many practical scenarios, it is desirable to structure codewords in such a way that they can be divided into two parts: an information part containing k unchanged information bits, and a redundancy part with $n - k$ symbols. A linear code with this structure is said to be in systematic form. The specific alignment of the information sequence within the codeword is not important; what matters is that the information sequence remains unaltered in the codeword. For convention, the information sequence is usually assumed to be right-aligned in the codeword. Under this assumption, a linear block code in systematic form is defined by a generator matrix $\mathbf{G}$ of the form $\mathbf{G} = [\mathbf{P} \mid \mathbf{I}_k]$ where $\mathbf{P}$ is a $k \times (n-k)$ matrix. Similarly, the parity-check matrix of a linear block code in systematic form $\mathbf{H}$ takes the form $\mathbf{H} = [\mathbf{I}_{n-k} \mid \mathbf{P}^{\mathrm{T}}]$.

2.1.2 Error Detection with Linear Block Codes

When a codeword $\mathbf{v} \in \mathcal{C}$ is transmitted over a binary-input binary-output channel, it results in a received vector $\mathbf{r}$. Due to channel noise, $\mathbf{r}$ and $\mathbf{v}$ may differ in some positions. To describe this scenario, an error pattern $\mathbf{e}$ is defined. The received vector $\mathbf{r}$ can be expressed as $\mathbf{r} = \mathbf{v} + \mathbf{e}$. Therefore, $e_j = 1$ for $r_j \neq v_j$ and $e_j = 0$ for $r_j = v_j$ with $j = 0, \cdots, n-1$. At the receiver, neither $\mathbf{v}$ nor $\mathbf{e}$ is known. The decoder's task is to first detect if there are errors in $\mathbf{r}$. Once the presence of errors is detected, the decoder can eventually estimate the error pattern $\mathbf{e}$ and then, the estimated transmitted codeword is computed as $\mathbf{v}^* = \mathbf{r} + \mathbf{e}^*$, where $\mathbf{e}^*$ is the estimate of $\mathbf{e}$.

Definition 2.4 (Syndrome)**.** *Let* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *be the parity-check matrix for a [n,k] code* $\mathcal{C}$ *and let* $\mathbf{v}$ *be a generic n-tuple over* $\mathbb{F}_2$*. The syndrome of* $\mathbf{v}$ *is denoted and defined as* $\mathbf{s} = \mathbf{H} \cdot \mathbf{v}^T \in \mathbb{F}_2^{n-k}$*.*

It is noteworthy that $\mathbf{v}$ belongs to the code $\mathcal{C}$ if and only if $\mathbf{H} \cdot \mathbf{v}^T = 0$.

Specifically, if $\mathbf{s} \neq \mathbf{0}$, it indicates that $\mathbf{v}$ is not a codeword in $\mathcal{C}$, and errors are detected. If $\mathbf{s} = \mathbf{0}$, the decoder assumes $\mathbf{r}$ is error-free and accepts it as the transmitted codeword. However, if $\mathbf{r}$ is a codeword but differs from $\mathbf{v}$, the decoder makes a decoding error. This happens when $\mathbf{e}$ transforms $\mathbf{v}$ into another codeword in $\mathcal{C}$ and this occurs when $\mathbf{e}$ is identical to a nonzero codeword in $\mathcal{C}$, known as an undetectable error pattern.

2.1.3 Error Correction with Linear Block Codes

Clearly, error detection serves as a prerequisite for error correction. For binary codes, error detection enables their correction, whereas, for non-binary codes error detection is the initial preparatory step for subsequent error correction. To formally assess the error-correcting capacity of a linear block code, it is first necessary to define what is meant by norm function.

Definition 2.5 (Norm function)**.** *A norm function $\|\cdot\|$ on a vector space V is a map that assigns a real value to each vector in V and that meets the properties of positivity, homothety and triangular inequality.*

The concept of a norm function ($\|\cdot\| : V \to \mathbb{R}$), depends on the specific vector space and the underlying field on which it is defined. When considering the vector space $V = \mathcal{R}$, composed of the 2^n n-tuples over the finite field $\mathbb{F}_2$, a common choice is the Hamming norm. The Hamming norm for a vector is quite straightforward - it is the count of non-zero elements within the vector.

Regarding the properties of a norm function, it is important to note that the Hamming norm possesses all of them. Firstly, it is is always non-negative, meaning that $\|\mathbf{v}\| \geqslant 0$ for any vector $\mathbf{v} \in V$, and it equals zero only when the vector is the null vector. Additionally, the Hamming norm satisfies the property of homothety. This means that when we multiply a vector by a scalar c (which is either 0 or 1 in $\mathbb{F}_2$), we find that $\|c \cdot \mathbf{v}\| = c \cdot \|\mathbf{v}\|$. Lastly, the Hamming norm also adheres the triangle inequality. In fact, for any pair

of vectors $\mathbf{v}$ and $\mathbf{w} \in V$, it holds true that $\|\mathbf{v} + \mathbf{w}\| \leqslant \|\mathbf{v}\| + \|\mathbf{w}\|$.

Definition 2.6 (Metric or Distance function)**.** *A metric function $d(\cdot)$ on a vector space V is a map that assigns a real value to a pair of vector in V and that meets the properties of positivity, symmetry and triangular inequality.*

Given a normed vector space (equipped with a norm function), it is possible to define the metric function induced by the norm as $d(v, w) = \|v - w\|$. This means that every normed vector space is also a metric space (equipped with a metric), while, in general, the opposite is not true.

Regarding the Hamming norm, based on what has been discussed so far, it is evident that it induces a metric function known as the Hamming distance. Specifically, the Hamming distance between two n-tuples, denoted as $\mathbf{u}$ and $\mathbf{v}$, is defined as the number of positions at which $\mathbf{u}$ and $\mathbf{v}$ differ, therefore, it is equal to the Hamming weight of their sum. It is important to emphasize that the Hamming distance, which is the metric induced by the Hamming norm, satisfies all the functional properties required to effectively consider it as a metric. Once the concept of a metric is defined, we are ready to introduce a crucial property of linear block codes, namely the minimum distance.

Definition 2.7 (Minimum Distance)**.** *Let $\mathcal{C}$ be a [n,k] linear code over $\mathcal{R}$ and let w be a norm on $\mathcal{R}$. The minimum distance of $\mathcal{C}$ is*

$$d = \min_{\mathbf{u},\ \mathbf{v} \in \mathcal{C},\ \mathbf{u} \neq \mathbf{v}} w(\mathbf{u} - \mathbf{v}).$$

The minimum distance of a code holds significant importance as it plays a pivotal role in determining the code's error-correcting capabilities. When we delve into the Hamming norm, the concept of the minimum distance revolves around identifying the shortest distance between codewords within the code. It goes without saying that a high minimum distance is a desirable attribute for a code, as it substantially diminishes the likelihood of misinterpretation when decoding the transmitted codeword.

According to the relationship between Hamming norm, Hamming metric and Hamming weight, it holds that:

$$d = \min_{\mathbf{u},\mathbf{v}\in\mathcal{C},\mathbf{u}\neq\mathbf{v}} w(\mathbf{u}-\mathbf{v}) = \min_{\mathbf{u},\mathbf{v}\in\mathcal{C},\mathbf{u}\neq\mathbf{v}} d(\mathbf{u},\mathbf{v}) = \min_{\mathbf{u},\mathbf{v}\in\mathcal{C},\mathbf{u}\neq\mathbf{v}} \mathrm{wt}(\mathbf{u}+\mathbf{v}) = \min_{\mathbf{x}\in\mathcal{C},\mathbf{x}\neq 0} \mathrm{wt}(\mathbf{x})$$

where $\mathbf{x} = \mathbf{v} + \mathbf{w} \in \mathcal{C}$ is a codeword according to the linearity of the code.

For $0 \leqslant i \leqslant n$, let A_i represent the number of codewords in $\mathcal{C}$ with a weight of i. The sequence of numbers $A_0, A_1, \ldots, A_n$ is referred to as the weight distribution of $\mathcal{C}$. The weight distribution of a block linear code $\mathcal{C}$ is closely associated with the parity-check matrix $\mathbf{H}$ of the code.

Theorem 2.1. *For an [n,k] linear block code $\mathcal{C}$ represented by the null space of a parity-check matrix $\mathbf{H}$, if there are no $d-1$ or fewer columns in $\mathbf{H}$ that sum to a zero vector, the minimum distance (or weight) of $\mathcal{C}$ is at least d. In other words, a linear code has distance d if and only if any $d-1$ columns of the parity check matrix are linearly independent, and there exist d columns that are linearly dependent.*

Proof. Consider a $[n,k]$ code $\mathcal{C}$ with a minimum distance of d. By contradiction, assume there are $d-1$ linearly dependent columns in $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Let $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{d-1}$ be these $d-1$ linearly dependent columns. Now, let's take a binary vector $\mathbf{x} \in \mathbb{F}_2^n$ with ones in entries corresponding to columns $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{d-1}$. Since $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{d-1}$ are linearly dependent for the initial assumption, it follows that $\mathbf{H}\cdot\mathbf{x}^T = \mathbf{0}$. Therefore, $\mathbf{x}$ is a codeword of $\mathcal{C}$ with a weight of $d-1$. This contradicts the fact that the minimum distance of $\mathcal{C}$ is d. Therefore, any $d-1$ columns must be linearly independent. Finally, assuming that the minimum distance is d, there must exist a binary vector $\mathbf{y} \in \mathcal{C}$ with a weight of d. Since $\mathbf{y}$ is a codeword, it follows that $\mathbf{H}\cdot\mathbf{y}^T = \mathbf{0}$. Hence, we obtain a linear combination of the corresponding d columns in the parity-check matrix $\mathbf{H}$ that equals $\mathbf{0}$. Hence, there must exist d linearly dependent columns. □

Theorem 2.1 guarantees that the minimum weight (or minimum distance)

of an [n,k] linear block code $\mathcal{C}$ with a parity-check matrix $\mathbf{H}$ is equal to the smallest number of columns in $\mathbf{H}$ whose vector sum is a zero vector. Specifically, it gives a lower bound on the minimum distance (or weight) of a linear block code. In general, it is very hard to determine the exact minimum distance (or weight) of a linear block code; however, it is much easier to give a lower bound on its minimum distance (or weight).

2.1.4 Decoding of Linear Block Codes

When a codeword $\mathbf{v} \in \mathcal{C}$ is transmitted, denoting $\mathbf{r}$ as the received vector, maximum-likelihood decoding (MLD) aims to decode $\mathbf{r}$ into a codeword $\mathbf{v}$ that maximizes the conditional a-posteriori probability $P(\mathbf{r}|\mathbf{v})$. In the case of a Binary Symmetric Channel (BSC), this corresponds to finding the codeword $\mathbf{v}$ that minimizes the Hamming distance $d(\mathbf{r}, \mathbf{v})$ between $\mathbf{r}$ and $\mathbf{v}$. This is known as minimum-distance or nearest-neighbor decoding. In minimum-distance decoding, the decoder must calculate the distance between $\mathbf{r}$ and every codeword in $\mathcal{C}$ and select a codeword $\mathbf{v}$ (which may not be unique) that is closest to $\mathbf{r}$. This process is referred to as complete error-correction decoding and requires evaluating the distance between $\mathbf{r}$ and all 2^k codewords in $\mathcal{C}$. However, for large values of k, implementing this complete decoder becomes practically infeasible.

Nonetheless, for many linear block codes, efficient algorithms have been developed for incomplete error-correction decoding, which achieves good performance with significantly reduced decoding complexity. Regardless of the specific codeword transmitted over a noisy channel, the received vector $\mathbf{r}$ can be one of the 2^n n-tuples over $\mathbb{F}_2$. The decoding scheme employed at the decoder essentially divides the vector space $V = \mathcal{R}$, comprising all n-tuples over $\mathbb{F}_2$, into 2^k regions. Each region contains one exclusive codeword from $\mathcal{C}$, and the goal of decoding is to identify the region that contains the received vector $\mathbf{r}$ and decode it into the appropriate codeword $\mathbf{v}$. These regions are known as decoding regions.

An algebraic method for partitioning 2^n received vectors into 2^k decoding regions involves constructing a so-called standard array for a $[n,k]$ linear block code. This $2^{(n-k)} \times 2^k$ array contains the 2^k codewords in its first row. Subsequent rows are generated one at a time by adding a selected vector $\mathbf{e}_j$, not present in the previous rows, to each codeword $\mathbf{v}_i$ in the top row and placing the resulting sum $\mathbf{e}_j + \mathbf{v}_i$ beneath $\mathbf{v}_i$. The choice of $\mathbf{e}_j$ is made to ensure no repetitions for vectors in the array. This process continues until no more vectors can be chosen from the vector space V, resulting in a standard array for $\mathcal{C}$, where each row corresponds to a coset, and the first element serves as the coset leader.

Based on the structure of the standard array, the sum of two vectors within the same coset yields a codeword. Moreover, there are no duplicated vectors in the array, and every n-tuples in the vector space V appears exactly once in the array. Additionally, all the vectors in a coset share the same syndrome, which is the syndrome of the coset leader. In other words, $(\mathbf{e}_j + \mathbf{v}_i) \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{e}_j \cdot \mathbf{H}^{\mathrm{T}}$ (since $\mathbf{v}_i \cdot \mathbf{H}^{\mathrm{T}} = 0$) and different cosets have different syndromes.

Consequently, there is a direct one-to-one correspondence between coset leaders and syndromes. Each column within the array functions as a decoding region, and the success of the decoding process relies on the accurate matching of the received vector with the transmitted codeword and its corresponding coset leader. To achieve optimal decoding, the choice of coset leaders should be guided by the most probable error patterns for the specific channel. In other words, optimality is achieved by considering the most probable error patterns for that particular type of channel as coset leaders (correctable errors).

In case of considering the Binary Symmetric Channels (BSCs), error patterns with minimal weight (fewer errors) are more probable. Therefore, selecting error patterns of this type makes minimum-distance decoding (MLD) highly effective, thus leading to the construction of an optimal standard array for the code $\mathcal{C}$.

At this point, all the preliminary concepts required to define the error-correcting capability of the code have been established. Specifically, let's consider an $[n,k]$ linear block code $\mathcal{C}$ with a parity-check matrix $\mathbf{H}$ and minimum distance d. It can be shown that all the n-tuples over $\mathbb{F}_2$ of weight $\delta = \lfloor \frac{d-1}{2} \rfloor$ or less can be used as coset leaders in an optimal standard array for $\mathcal{C}$. In some cases, to complete the standard array it is possible to consider some (but not all) cosets leader (correctable errors) of weight $\delta + 1$. Therefore, for a linear code $\mathcal{C}$ with minimum distance d, any error pattern with δ or fewer errors is guaranteed correctable (i.e., resulting in correct decoding), but not all the error patterns with $\delta + 1$ or more errors. The parameter δ is called the error-correction capability of $\mathcal{C}$. We say that $\mathcal{C}$ is capable of correcting δ or fewer random errors and $\mathcal{C}$ is called a δ-error-correcting code.

Decoding an $[n,k]$ linear block code $\mathcal{C}$ using an optimal standard array can be challenging due to the large memory required to store 2^n n-tuples. However, this complexity can be significantly reduced by recognizing that coset leaders encompass all correctable error patterns and that there is a direct correspondence between $(n-k)$-tuple syndromes and coset leaders.

To simplify decoding, a table with two columns is created, containing 2^{n-k} coset leaders (correctable error patterns) in one column and their corresponding syndromes in the other. The decoding process for the vector $\mathbf{r}$ begins with the computation of its syndrome $\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^{\mathrm{T}}$. The goal is to find the coset leader $\mathbf{e}$ in the table whose syndrome matches $\mathbf{s}$. This $\mathbf{e}$ is assumed to represent the error pattern from the channel noise, and $\mathbf{r}$ is then decoded into the codeword $\mathbf{v} = \mathbf{r} + \mathbf{e}$. This approach is known as syndrome decoding or table-look-up decoding and significantly reduces decoder complexity compared to standard-array-based decoding. For a long code with large $n-k$, a complete table-look-up decoder is still very complex, requiring a very large memory to store the look-up table. If we limit ourself to correcting only the error patterns guaranteed by the error-correcting capability δ of the code, the size of

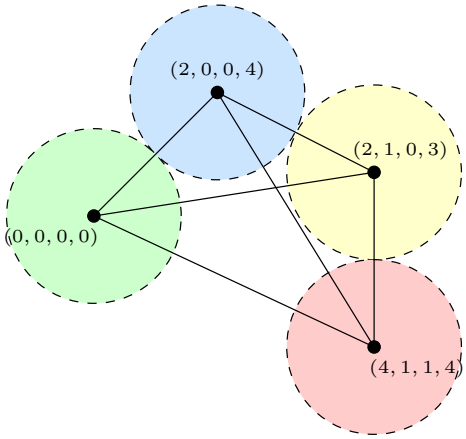

Figure 2.1: Example of 4 distinct codewords alongside their respective decoding spheres. Noticeably, each decoding sphere has a radius matching the code's correction capability, which is, in turn, intricately tied to the code's minimum distance

the look-up table can be further reduced. The new table consists of only

$$N_\delta = \binom{n}{0} + \binom{n}{1} + \cdots \binom{n}{\delta}$$

correctable error patterns guaranteed by the minimum distance d of the code.

With this above partial table-look-up decoding, the number of errors to be decoded is bounded by the error-correcting capability, therefore, this is called bound-distance decoding. To visually grasp the concept of bound distance decoding, refer to Figure 2.1. In this context, the term "decoding region" denotes the spatial vicinity around each codeword, wherein the decoding procedure is anticipated to yield successful results. Going into more details, if the error pattern is such that from one codeword, we move to a word that is inside the decoding region associated with another codeword, the decoding process fails. Restricting decoding to the minimum distance of the code means ensuring that if the weight of the error pattern is less than the correction capability of the code, then the error pattern, whatever it may be, is such that from one codeword, we still end up with a word that is within the decoding region of that codeword.

2.2 Cyclic codes

Cyclic codes form a very special type of linear block codes. They have encoding advantage over many other codes since encoding can be implemented with simple shift registers with feedback connections.

Definition 2.8 (Cyclic code)**.** *An [n, k] linear block code $\mathcal{C}$ is said to be cyclic if the cyclic-shift of each codeword in $\mathcal{C}$ is also a codeword in $\mathcal{C}$. In other words, a linear block code is said to be cyclic if the 2^k codewords form a k-dimensional subspace closed under the operation of cyclic-shift within the vector space V of 2^n n-tuples in $\mathbb{F}_2$.*

When it comes to cyclic codes, it is generally preferred to work with the polynomial representation rather than the vectorial one. A codeword $\mathbf{v} \in \mathcal{C}$ is represented by a polynomial over $\mathbb{F}_2$ of degree $n-1$ or less with components of $\mathbf{v}$ as coefficients. This polynomial is called code polynomial. Obviously, there is a one-to-one correspondence between codewords and code polynomials, therefore, a $[n,k]$ cyclic code consists of 2^k code polynomials. In an $[n,k]$ cyclic code $\mathcal{C}$, every nonzero code polynomial has degree at least $n-k$ but not greater than $n-1$. There exists one and only one code polynomial $g(x)$ of degree $n-k$ of the following form:

$$g(x) = 1 + g_1 \cdot x + g_2 \cdot x^2 + \cdots g_{n-k-1} \cdot x^{n-k-1} + x^{n-k}.$$

The polynomial $g(x)$ is said to be the generator polynomial of the $[n,k]$ cyclic code $\mathcal{C}$. As is evident from the name, the generator polynomial of a cyclic code is the "polynomial version" of the generator matrix of a linear block code. Therefore, all operations that involve the generator matrix in a linear block code can also be defined in terms of the generator polynomial while retaining the same meaning. In particular, concerning the encoding rule, it follows that:

$$v(x) = m(x)g(x)$$

where $m(x) = m_0 + m_1 \cdot x + \cdots m_{k-1}x^{k-1}$ is a polynomial over $\mathbb{F}_2$ of degree $k-1$ or less and $\mathbf{m} = (m_0, m_1, \cdots m_{k-1})$ is the message to be encoded.

Clearly, the generator polynomial and the generator matrix are deeply linked to each other. Specifically, the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ of the [n,k] cyclic code is simply obtained by using the n-tuple representation of the generating polynomial $g(x)$ as the first row and its $k-1$ circular shifts to the right like the remaining $k-1$ rows. A crucial property of the generator polynomial $g(x)$ of an [n,k] cyclic code $\mathcal{C}$ is that it divides $x^n + 1$. Consequently, $x^n + 1$ can be expressed as the following product:

$$x^n + 1 = g(x)f(x)$$

where $f(x)$ is a polynomial of degree k over $\mathbb{F}_2$. The reciprocal polynomial of $f(x)$ denoted by $h(x) = x^k \cdot f(x^{-1})$ is called parity-check polynomial of $\mathcal{C}$. Similar to what we observed with the generator polynomial and the generator matrix, the check-parity polynomial similarly defines the check-parity matrix.

2.2.1 Systematic Encoding of Cyclic codes

Systematic encoding of an [n,k] cyclic code using a generator polynomial $g(x)$ is straightforward. Given a message $\mathbf{m} = (m_0, m_1, \cdots, m_{k-1})$, it is possible to deduce the message polynomial $m(x) = m_0 + m_1 \cdot x + \cdots + m_{k-1} \cdot x^{k-1}$. This can be multiplied by x^{n-k}, resulting in $x^{n-k} \cdot m(x)$, a polynomial of maximum degree $k-1+n-k = n-1$. Dividing $x^{n-k} \cdot m(x)$ by the generator polynomial $g(x)$, we obtain:

$$x^{n-k} \cdot m(x) = a(x)g(x) + b(x)$$

where $a(x)$ is the quotient and $b(x)$ is the remainder. The degree of $b(X)$ is $n-k-1$ or less, and it takes the form $b(X) = b_0 + b_1x + \cdots + b_{n-k-1}x^{n-k-1}$. Since $b(x) + x^{n-k} \cdot m(x) = a(x)g(x)$, it means that $b(x) + x^{n-k} \cdot m(x)$ is divisible by $g(x)$, therefore, $b(x) + x^{n-k} \cdot m(x)$ is a code polynomial. At this

point we can observe that the n-tuple representation of the code polynomial $b(x) + x^{n-k} \cdot m(x)$ is in systematic form, specifically:

$$(b_0, b_1, \cdots b_{n-k-1}, m_0, m_1, \cdots, m_{k-1})$$

where the $n - k$ parity check bits are the coefficients of the remainder $b(x)$.

2.2.2 Decoding of Cyclic codes

For the decoding of cyclic codes, being them a subset of linear block codes, the same procedures described in subsection 2.1.4 can be applied. However, as previously mentioned, in the context of cyclic codes, it is preferred to work with the polynomial representation. Therefore, it is necessary to define the polynomial equivalent of the syndrome previously defined in the vector domain. Let $r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1}$ represent the received polynomial. As usual, $r(x)$ can be expressed as the sum of a transmitted polynomial $v(x)$ (code polynomial) and an error polynomial $e(x)$, namely $r(x) = v(x) + e(x)$.

The initial step in decoding $r(x)$ involves calculating its syndrome, denoted as $s(x)$, which is determined by the remainder when $r(x)$ is divided by the generator polynomial $g(x)$ of code $\mathcal{C}$. If $s(x) = 0$, the receiver accepts $r(x)$ as the transmitted code polynomial. If $s(x) \neq 0$, this indicates the presence of transmission errors. In such cases, error patterns can be detected and eventually corrected, depending on the code's error-correcting capabilities. It is noteworthy that the maximum degree of the syndrome is $n - k - 1$, which means that in the vector domain, it consists of $n - k$ coefficients, consistent with the syndrome definition provided in the vector domain. It's important to emphasize that in this case, the definition of syndrome changes to accommodate the context of polynomials. However, the definition is tailored to ensure that the syndrome retains the same meaning.

2.2.3 Shortened Cyclic Codes

Let us consider a systematic [n,k] cyclic code $\mathcal{C}$. Among all the 2^k codewords, assuming ℓ is a non-negative integer less than k ($\ell < k$), it is possible to consider only the set of code polynomials that have ℓ leading high-order information digits as zeros, namely, $v_{n-\ell}, \cdots, v_{n-2}, v_{n-1}$. This set comprises $2^{k-\ell}$ code polynomials. By eliminating the ℓ zero information digits from each of these code polynomials, we obtain a set of $2^{k-\ell}$ polynomials over $\mathbb{F}_2$ with maximum degree $n-\ell-1$. These $2^{k-\ell}$ shortened polynomials form an $[n-\ell, k-\ell]$ linear block code known as shortened cyclic code, which, unlike the original code is not cyclic. A shortened cyclic code offers at least the same error-correction capability as the original code from which it's derived because in the considered codewords, we are only removing 0s. Therefore, the minimum Hamming weight which is equivalent to the minimum Hamming distance, as discussed in subsection 2.1.3, remains unchanged.

2.2.4 Quasi-Cyclic codes

Let s and n be two positive integers and consider the $(s \cdot n)$-tuple over $\mathbb{F}_2$, denoted and defined by $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2 \cdots \mathbf{c}_s)$, that consists of s sections of n bits each. For $1 \leqslant j \leqslant s$, the j-th section of $\mathbf{c}$ is a n-tuple over $\mathbb{F}_2$, i.e., $\mathbf{c}_j = (c_{j,0}, c_{j,1}, \cdots, c_{j,n-1})$. Let $\mathbf{c}_j^{(1)}$ be the n-tuple over $\mathbb{F}_2$ obtained by cyclically shifting each component of $\mathbf{c}_j$ one place to the right. We call $\mathbf{c}_j^{(1)}$ the right cyclic-shift of $\mathbf{c}_j$, namely $\mathbf{c}_j^{(1)} = (c_{j,n-1}, c_{j,0}, \cdots, c_{j,n-2})$. Let $\mathbf{c}^{(1)}$ be the $(s \cdot n)$-tuple over $\mathbb{F}_2$ obtained by cyclically shifting each section of $\mathbf{c}$ one place to the right, formally:

$$\mathbf{c}^{(1)} = (\mathbf{c}_1^{(1)} \mathbf{c}_2^{(1)} \cdots \mathbf{c}_s^{(1)}).$$

Definition 2.9. *(Quasi Cyclic code) Let n, k and s be positive integers such that $k < s \cdot n$. Let $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \cdots \mathbf{c}_s)$ be a vector in $\mathbb{F}_2^{sn}$ (s following n-tuple*

over $\mathbb{F}_2$*). An [sn, k, d] linear code* $\mathcal{C}$ *is Quasi-Cyclic (QC) of index s if for any* $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \cdots \mathbf{c}_s) \in \mathcal{C}$*, the vector* $\mathbf{c}^{(1)}$ *obtained after applying a simultaneous circular shift to every block, namely,* $\mathbf{c}_1, \mathbf{c}_2, \cdots \mathbf{c}_s$ *is also a codeword.*

More formally, by considering each block $\mathbf{c}_i$ as a polynomial in $\mathcal{R}$, the code $\mathcal{C}$ is a Quasi-Cyclic code of index s if for any $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \cdots \mathbf{c}_s) \in \mathcal{C}$ it holds that $(x \cdot \mathbf{c}_1, x \cdot \mathbf{c}_2, \cdots x \cdot \mathbf{c}_s) \in \mathcal{C}$. Specifically, if $s = 1$, the QC code $\mathcal{C}$ is a also a cyclic code. Therefore, cyclic codes form a subclass of QC codes. In general, it can be stated that Quasi-Cyclic codes are codes that are cyclic section by section (section-wise). Among QC codes, Systematic Quasi-Cyclic (QC) codes deserve special mention due to their structured format. This structure extends to the parity-check matrix, making them worthy of further exploration.

Definition 2.10. *(Systematic Quasi-Cyclic codes) A systematic QC [sn, n] code of index s and rate* $1/s$ *is a quasi-cyclic code with an* $(s-1)n \times sn$ *parity-check matrix of the form:*

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & 0 & \cdots & 0 & \mathbf{A}_0 \\ 0 & \mathbf{I}_n & & 0 & \mathbf{A}_1 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \mathbf{I}_n & \mathbf{A}_{s-2} \end{pmatrix}.$$

where $\mathbf{A}_0$, $\mathbf{A}_1$ and $\mathbf{A}_{s-2}$ are $n \times n$ circulant matrix as the ones described in chapter 1.

2.3 Brief generalization to non-binary codes

In this section, our aim is to provide a comprehensive overview of the fundamental properties of non-binary codes, with a specific focus on how they relate to the binary codes already discussed. Up until this point, in both section 2.1 and section 2.2, our discussions have primarily centered around block codes with symbols derived from the binary field $\mathbb{F}_2$. Now, we extend our scope to

encompass block codes with symbols originating from non-binary fields. These codes, known as q-ary block codes, follow a similar construction approach to binary block codes. A q-ary block code $[n, k]$ has length n and comprises q^k codewords. In the context of a q-ary block code $[n, k]$, a message consists of k informative symbols, each drawn from $\mathbb{F}_q$.

Definition 2.11. *(Non-Binary block code) A q-ary block code of length n with q^k codewords is called a q-ary block linear code $[n, k]$ if and only if its q^k codewords collectively form a vector subspace within the vector space of all q^n n-tuples with coefficients in $\mathbb{F}_q$.*

A q-ary block linear code $[n, k]$ is defined by a generator matrix $\mathbf{G}$ of dimensions $k \times n$ with coefficients in $\mathbb{F}_q$ and a parity-check matrix $\mathbf{H}$ of dimensions $(n - k) \times n$ with coefficients in $\mathbb{F}_q$. In comparison to binary codes, the principal distinction lies in the fact that the elements of these matrices belong to $\mathbb{F}_q$ instead of $\mathbb{F}_2$. The encoding and decoding of a q-ary block linear code $[n, k]$ are exactly the same as for a binary code, with the sole distinction that the operations are conducted in $\mathbb{F}_q$ instead of $\mathbb{F}_2$. Moreover, concerning the family of cyclic codes, it holds that a q-ary cyclic code $[n,k]$ is generated by a generator polynomial of degree $n - k$ with coefficients in $\mathbb{F}_q$ defined as:

$$g(x) = g_0 + g_1 x + \cdot g_{n-k-1} x^{n-k-1} + x^{n-k}$$

where g_i for $i = 0, \cdots n - k - 1 \in \mathbb{F}_q$.

Chapter 3

Codes of interest in HQC

After offering a concise overview of the essential characteristics of linear block codes (in section 2.1) and introducing cyclic codes as a subset of linear block codes (in section 2.2), along with a generalization of the discussion to the non-binary case (in section 2.3), this chapter's goal is to explore different code types employed in Hamming Quasi-Cyclic (HQC).

3.1 Repetition codes

A repetition code (REP) denoted by $\mathcal{C}_{rep}$ over $\mathbb{F}_2$ of length n is a binary [n,1] linear code with a single information bit ($k = 1$). The code is simply obtained by repeating a single information bit n times. Therefore, it consists of only two codewords, namely the all-zero codeword $(0, \cdots, 0)$ and the all-one codeword, $(1, \cdots, 1)$. Obviously, its generator matrix is $\mathbf{G}_{rep} = (111 \cdots 1)$. In this way:

- $\mathbf{u} = 0 \rightarrow \mathbf{v} = \mathbf{u} \cdot \mathbf{G}_{rep} = (0, \cdots, 0)$
- $\mathbf{u} = 1 \rightarrow \mathbf{v} = \mathbf{u} \cdot \mathbf{G}_{rep} = (1, \cdots, 1)$

The minimum distance of a REP code, denoted as d, is exactly n. Therefore, the corrective power of a REP code is:

$$\delta = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{n-1}{2} \right\rfloor$$

3.2 Binary BCH codes

BCH codes (Bose-Chaudhuri-Hocquenghem) belong to the cyclic code family (described in section 2.2). Due to their cyclic nature, their construction relies on defining the generator polynomial $g(x)$, specifically by defining its roots.

Theorem 3.1. *For every positive integer $m \geqslant 3$ and every positive integer t less than or equal to 2^{m-1}, there exists a binary BCH code with a block length of $2^m - 1$, a minimum distance of at least $2t + 1$, and a maximum of $m \cdot t$ redundancy symbols.*

This particular BCH code, usually denoted as $[n, k, \delta]$, is capable of correcting up to $\delta = t$ errors, hence, it is referred to as a t-error-correcting BCH code. The construction of a BCH code capable of correcting t errors starts with defining an extended Galois field denoted as $\mathbb{F}_{2^m}$. Let α be a primitive element of $\mathbb{F}_{2^m}$, meaning α can generate all nonzero elements of $\mathbb{F}_{2^m}$ with its powers. The generator polynomial $g(x)$ of the BCH code of length $2^m - 1$ is the minimal degree polynomial with coefficients in $\mathbb{F}_2$ whose roots are the $2t$ consecutive powers of α, namely α, α^2, $\cdots$, α^{2t}.

Definition 3.1. *(minimal polynomial) Let β be an element of the extended Galois field $\mathbb{F}_{2^m}$ (an extension of $\mathbb{F}_2$). The monic polynomial with coefficients in $\mathbb{F}_2$ that has the minimum degree and has β as its roots is said to be the minimal polynomial of β and it is denoted by $\phi(x)$.*

For $1 \leqslant i \leqslant 2t$, let $\phi_i(x)$ be the minimal polynomial of α^i. Then, the generator polynomial of the t-error correcting binary BCH code of length 2^m-1 is given by the least common multiple (LCM) of $\phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x)$, i.e.,

$$g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x)\}.$$

Once the generator polynomial of a BCH code is defined, as it is cyclic, the code itself is completely defined.

3.2.1 Decoding of binary BCH codes

Consider a $[n,\ k,\ \delta]$ binary BCH code with $n = 2^m - 1$ and assume that a code polynomial $v(x)$ is transmitted over a Binary Symmetric Channel. The received polynomial is denoted as $r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1}$. Error locations are identified through $e(x) = e_0 + e_1 x + \ldots + e_{n-1}x^{n-1}$ which is known as the error polynomial. Specifically, if $e_i = 1$, it means the occurrence of an error at that particular location. Then we can write:

$$r(x) = v(x) + e(x).$$

As usual, the first step in decoding involves computing the syndrome of $r(x)$. Since, $v(x) = m(x)g(x)$, when $x = \alpha^i$, it follows that $v(\alpha^i) = m(\alpha^i)g(\alpha^i)$. Given that $g(\alpha^i) = 0$ for $i = 1, \cdots 2t$, it consequently holds that $v(\alpha^i) = 0$. Therefore, $r(\alpha^i) = v(\alpha^i) + e(\alpha^i) = e(\alpha^i)$. The equation $v(\alpha^i) = 0$ can be represented in matrix form, as:

$$(v_0, v_1, \cdot, v_{2^m-2}) \cdots \begin{pmatrix} 1 \\ \alpha^i \\ \vdots \\ \alpha^{(2^m-2)\cdot i} \end{pmatrix} = 0$$

This matrix representation allow us to define the structure of the parity-check matrix $\mathbf{H}$ for a BCH code, which is crucial for syndrome computation. Specifically:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(2^m-2)} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{2^m-2} \\ 1 & \alpha^3 & (\alpha^3)^2 & \cdots & (\alpha^3)^{2^m-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \cdots & (\alpha^{2t})^{2^m-2} \end{pmatrix}$$

This matrix is a Vandermonde matrix.

Definition 3.2. *(Vandermonde matrix) In linear algebra, a Vandermonde matrix is a matrix whose rows (or columns) have elements in a geometric progression starting from 1, so that the element in position (i,j) is* ${\alpha_i}^{j-1}$.

If we consider any value of $d \leqslant 2t$, it turns out that every $d \times d$ submatrix of $\mathbf{H}$ is still a Vandermonde matrix with a non-zero determinant indicating that its columns are linearly independent. According to Theorem 2.1, when calculating the minimum distance of a code, we need to determine the minimum number of columns whose sum equals zero. Clearly, in this case, taking a submatrix of $\mathbf{H}$ of size $d \times d$ with $d \leqslant 2t$, columns do not sum to zero because they are linearly independent. This implies that to obtain a zero sum, a number of columns at least equal to $2t + 1$ is needed. Thus, the minimum distance of this BCH code is at least $2t + 1$ which is referred to as the designed minimum distance of the BCH code. The actual minimum distance can be greater than $2t + 1$; therefore, $2t + 1$ serves as a lower bound for the minimum distance of a BCH code.

Once we have defined the parity-check matrix for a BCH code, we can determine the syndrome of $r(x)$ as:

$$\mathbf{S} = (S_1, S_2, \cdots, S_{2t}) = \mathbf{r} \cdot \mathbf{H}^{\mathrm{T}}.$$

For $1 \leqslant i \leqslant 2t$, it follows that the i-component of $\mathbf{S}$ is:

$$S_i = (r_0, r_1, \cdots, r_{2^m-2}) \cdot \begin{pmatrix} 1 \\ \alpha^i \\ \vdots \\ \alpha^{(2^m-2)\cdot i} \end{pmatrix} = r_0 + r_1\alpha^i + \cdots + r_{2^m-2}\alpha^{(2^m-2)} = r(\alpha^i).$$

Therefore, since $r(\alpha^i) = e(\alpha^i)$ it follows that:

$$S_i = e(\alpha^i).$$

Suppose that $e(x)$ has ν errors at location $j_1, j_2, \cdots j_\nu$, then

$$e(x) = x^{j_1} + x^{j_2} + \cdots x^{j_\nu},$$

we obtain the following set of equations, where $\alpha^{j_1}, \alpha^{j_2}, \cdots \alpha^{j_\nu}$ are unknown:

$$\begin{aligned} S_1 &= \alpha^{j_1} + \alpha^{j_2} + \cdots + \alpha^{j_\nu} \\ S_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \cdots + (\alpha^{j_\nu})^2 \\ &\vdots \\ S_{2t} &= (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + \cdots + (\alpha^{j_\nu})^{2t} \end{aligned}$$

The aim of a BCH decoding algorithm is to solve this system of equations. In particular, $S_1, S_2, \cdots, S_{2t}$ are known since the syndrome is calculated in reception. Specifically, by solving the system we determine $\alpha^{j_1}, \alpha^{j_2}, \cdots \alpha^{j_\nu}$ where the exponents are precisely the sought-after unknowns. Given that $\alpha^{j_1}, \alpha^{j_2}, \cdots \alpha^{j_\nu}$ allow us to identify the error positions, they are referred to as error-location numbers. For the sake of simplicity, let's ease the notation assuming $\beta_i = \alpha^{j_i}$. In this way, the equations above can be expressed as follows:

$$\begin{aligned} S_1 &= \beta_1 + \beta_2 + \cdots + \beta_\nu \\ S_2 &= (\beta_1)^2 + (\beta_2)^2 + \cdots + (\beta_\nu)^2 \\ &\vdots \\ S_{2t} &= (\beta_1)^{2t} + (\beta_2)^{2t} + \cdots + (\beta_\nu)^{2t} \end{aligned}$$

This is a non-linear system over a finite field. With the new notation, the unknowns are $\beta_l = \alpha^{j_l}$ with $1 \leqslant l \leqslant \nu$. Solving this system, being non-linear, is particularly challenging. In order to tackle the problem of solving the non-

linear system in a feasible manner, the following polynomial is introduced:

$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x) \cdots (1 + \beta_\nu x)$$

where $\sigma(x)$ is a polynomial of degree ν whose roots are $\beta_1^{-1}, \beta_2^{-1}, \cdots \beta_\nu^{-1}$.

Evaluating the construction of the polynomial $\sigma(x)$ we realize that it can be expressed as $\sigma(x) = \sigma_0 + \sigma_1 x + \cdots \sigma_\nu x^\nu$ where $\sigma_0 = 1$, $\sigma_1 = \beta_1 + \beta_2 + \cdots + \beta_\nu$, $\sigma_2 = \beta_1\beta_2 + \beta_1\beta_3 + \cdots + \beta_{\nu-1}\beta_\nu$, $\cdots$, $\sigma_\nu = \beta_1\beta_2 \cdots \beta_\nu$. The polynomial $\sigma(x)$ is said to be the error-location polynomial. At this point, it is possible to define an equivalent set of equations using the error-location polynomial, this new equations are known as Newton identities:

$$\begin{aligned}
&S_1 + \sigma_1 = 0 \\
&S_2 + \sigma_1 S_1 + 2\sigma_2 = 0 \\
&S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 = 0 \\
&\vdots \\
&S_\nu + \sigma_1 S_{\nu-1} + \sigma_2 S_{\nu-2} + \cdots + \sigma_{\nu-1} S_1 + \nu\sigma_\nu = 0 \\
&\vdots
\end{aligned}$$

By solving the previous system, it is possible to determine $\sigma_1, \sigma_2, ..., \sigma_\nu$, and thus the polynomial $\sigma(x)$. Once $\sigma(x)$ is determined, we evaluate its roots. The inverses of the roots of $\sigma(x)$ correspond to $\beta_1, \beta_2, \beta_3, \cdots, \beta_\nu$. From the relationship $\beta_i = \alpha^{j_i}$, we determine j_i, and consequently, the error pattern $e(x) = x^{j_1} + x^{j_2} + \cdots x^{j_\nu}$. By subtracting $e(x)$ from the received polynomial $r(x)$ (or adding using modulo-2 addition), we obtain the decoded codeword $v(x)$.

The main difficulty in this decoding algorithm lies in the resolution of the Newton identities. In fact, besides the inherent complexity of the problem, the solution of the Newton identities is not unique. Therefore, in general, there

will be more than one error pattern for which the coefficients of the polynomial $\sigma(x)$ satisfy these identities. To minimize the decoding error probability, we need to find the most probable error pattern. For the Binary Symmetric Channel (BSC), finding the most probable error pattern means determining the $\sigma(x)$ polynomial of the minimum degree whose coefficients satisfy the Newton identities. Indeed, the degree of the $\sigma(x)$ polynomial is equal to the number of errors in the codeword, so we reasonably assume that the error count is minimized. This can be achieved iteratively using an algorithm commonly referred to as the Berlekamp-Massey (BM) algorithm. This algorithm proceeds one equation at a time. Specifically, the algorithm determines the $\sigma(x)$ polynomial of minimal degree that satisfies the k-th Newton identity. Then, it checks if the polynomial also satisfies the subsequent equation, i.e., the $k+1$-th identity. If not, a corrective term known as "discrepancy" is introduced to make the $\sigma(x)$ polynomial a solution to the $k+1$-th Newton identity. We won't delve further into this algorithm here, but that is the basic idea.

3.3 Non-binary BCH codes

In HQC we are going to use Reed-Solomon (RS) codes. These codes are a specific case of q-ary BCH codes. Therefore, before delving into understanding RS codes, it is first necessary to generalize the discussion about binary BCH codes (in section 3.2) to the non-binary case. Let $\mathbb{F}_q^m$ be the extended Galois field of the prime field $\mathbb{F}_q$. Let α be a primitive element of $\mathbb{F}_q^m$. A BCH code of length $q^m - 1$ over $\mathbb{F}_q$, capable of correcting t errors, is a cyclic code generated by the polynomial of minimum degree, denoted as $g(x)$, over $\mathbb{F}_q$ which has $\alpha, \alpha^2, \cdots \alpha^{2t}$ and their conjugates as roots. For $1 \leqslant i \leqslant 2t$, let $\phi_i(x)$ be the minimum polynomial of α_i over $\mathbb{F}_q$. It turns out that:

$$g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x)\}.$$

The parity-check matrix **H** of a q-ary BCH code has the same structure of the parity-check matrix of a binary BCH code; therefore, it is a Vandermonde matrix. Since the parity-check matrix **H** is a Vandermonde matrix, as seen in the case of binary BCH codes (in section 3.2), it follows that the minimum distance of the code is at least $2t + 1$. To summarize, a q-ary BCH code capable of correcting t errors has a length of $n = q^m - 1$, a maximum number of redundancy symbols of $2mt$, at least dimension $q^m - 1 - 2mt$ and minimum distance of at least $2t + 1$.

3.3.1 Reed-Solomon codes

The most important and widely used class of q-ary BCH codes is represented by Reed-Solomon (RS) codes. Specifically, these can be considered as a special case of q-ary BCH codes when the code's construction field, denoted as $\mathbb{F}_{q^m}$, is the same as the symbol field, denoted as $\mathbb{F}_q$. This means that RS codes are a specialization of BCH q-ary codes in the case of $m = 1$. Let α be a primitive element of $\mathbb{F}_{q^m} = \mathbb{F}_q$, an RS code with a length of $q^m - 1 = q - 1$ over $\mathbb{F}_q$ and capable of correcting t errors, with $2t < q$, is a cyclic code generated by the minimal-degree polynomial $g(x)$ (the generator polynomial) over $\mathbb{F}_q$ that has $\alpha, \alpha^2, \ldots, \alpha^{2t}$ and their conjugates as roots. In specific terms, for an RS code, it can be shown that the minimum distance is exactly equal to the lower bound, i.e., $d = 2t + 1$. This means that the minimum distance of an RS code is one more than the number of redundancy symbols. A code of this type is referred to as maximum-distance-separable (MDS). In other words, such codes are optimal, meaning that given the same number of redundancy symbols, it's not possible to correct more errors. In summary, an RS code that corrects t errors over $\mathbb{F}_q$ has a length of $q - 1$, presents $2t$ redundancy symbols, has a dimension of $q - 1 - 2t$ and a minimum distance of $2t + 1$.

In all practical applications of RS codes, q is commonly chosen as a power of 2, let's say $q = 2^s$, and the code symbols are drawn from $\mathbb{F}_{2^s}$. If each code

symbol is represented by an s-tuple over $\mathbb{F}_2$, an RS code can be transmitted using binary modulation, such as BPSK. During the decoding process, every s received bits are grouped into a received symbol in $\mathbb{F}_{2^s}$. Subsequently, decoding is performed on the received symbol sequence.

3.3.2 Decoding Reed Solomon codes

To conclude this section, we will provide further insights into the decoding process of RS codes. Unlike binary codes, RS codes decoding introduce additional complexity due to their non-binary nature. In a binary code, once the error position is identified, automatic correction can be applied. However, in the non-binary scenario, after identifying the error position, direct correction is not possible because it remains uncertain which of the remaining $q-1$ symbols is the correct one. In contrast to decoding binary BCH codes, the decoding of RS codes requires the definition of not only the error locator polynomial, denoted as $\sigma(x)$, but also another polynomial called the error evaluator polynomial, represented as $Z_0(x)$.

Consider an RS code $[q-1,\ q-2t-1,\ 2t+1]$ over $\mathbb{F}_q$, designed for error correction on a Binary Symmetric Channel (BSC). Let's assume the transmitted codeword polynomial is $v(x) = v_0 + v_1 x + \cdots + v_{q-2}x^{q-2}$, and the received polynomial is $r(x) = r_0 + r_1 x + \cdots + r_{q-2}x^{q-2}$. Furthermore, the error pattern is represented by the following polynomial:

$$e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \cdots + e_{j_\nu}x^{j_\nu}$$

where $0 \leqslant j_1 < j_2 < \cdots < j_\nu < q-1$ are the error positions and $e_{j_1}, e_{j_2}, \cdots, e_{j_\nu}$ are the error values at positions $j_1, j_2, \cdots, j_\nu$.

We establish the set of syndromes as $S_1, S_2, \cdots S_{2t}$, defined as $S_i = r(\alpha^i)$, where α represents a primitive element in $\mathbb{F}_{2^m}$. As explained in the binary case, we can observe that $r(\alpha^i) = e(\alpha^i)$ since $v(\alpha^i) = 0$ (where $v(x)$ is the

code polynomial). This leads us to the following system of equations, where $\alpha^{j_1}, \alpha^{j_2}, \cdots, \alpha^{j_n u}$ are unknown:

$$\begin{aligned} S_1 &= e_{j_1}\alpha^{j_1} + e_{j_2}\alpha^{j_2} + \cdots + e_{j_\nu}\alpha^{j_\nu} \\ S_2 &= e_{j_1}(\alpha^{j_1})^2 + e_{j_2}(\alpha^{j_2})^2 + \cdots + e_{j_\nu}(\alpha^{j_\nu})^2 \\ &\vdots \\ S_{2t} &= e_{j_1}(\alpha^{j_1})^{2t} + e_{j_2}(\alpha^{j_2})^{2t} + \cdots + e_{j_\nu}(\alpha^{j_\nu})^{2t} \end{aligned}$$

The aim of a Reed-Solomon decoding algorithm is to solve this system of equations. To simplify the notation let's assume $\beta_i = \alpha^{j_i}$, then, it follows that:

$$\begin{aligned} S_1 &= e_{j_1}\beta_1 + e_{j_2}\beta_2 + \cdots + e_{j_\nu}\beta_\nu \\ S_2 &= e_{j_1}(\beta_1)^2 + e_{j_2}(\beta_2)^2 + \cdots + e_{j_\nu}(\beta_\nu)^2 \\ &\vdots \\ S_{2t} &= e_{j_1}(\beta_1)^{2t} + e_{j_2}(\beta_2)^{2t} + \cdots + e_{j_\nu}(\beta_\nu)^{2t} \end{aligned}$$

As discussed for the binary case we define the error location polynomial as:

$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)\cdots(1 + \beta_\nu x) = \sigma_0 + \sigma_1 x + \cdots \sigma_\nu x^\nu$$

where $\sigma(x)$ is a polynomial of degree ν whose roots are $\beta_1^{-1}, \beta_2^{-1}, \beta_3^{-1}, \cdots \beta_\nu^{-1}$. After retrieving the coefficients of $\sigma(x)$, we can compute the error positions. Define the error evaluator polynomial $Z_0(x)$ as:

$$Z_0(x) = S_1 + (S_2 + \sigma_1 S_1)x + \cdots + (S_0 + \sigma_1 S_{\nu-1} + \cdots + \sigma_{\nu-1} S_1)x^{\nu-1}.$$

The problem of decoding an RS code translates into evaluation of the error locator polynomial $\sigma(x)$ and the error evaluator polynomial $Z_0(x)$. These two

polynomials are closely linked by the so-called key equation:

$$\sigma(x)S(x) = Z_0(x) \bmod x^{2t}$$

where the polynomial $S(x)$ is a known polynomial of degree $2t-1$ with coefficients $S_1, S_2, \ldots, S_{2t}$.

Decoding RS codes involves resolving the key equation, and any method for this purpose serves as a decoding technique. When the error count in $e(x)$ remains within the code's correction capability, the key equation has two distinct solutions: $\sigma(x)$ and $Z_0(x)$. Notably, the degree of $\sigma(x)$ is lower than or equal to the degree of $Z_0(x)$, which, in turn, is either lower or at most equal to the code's correction capability. One widely used method for tackling the key equation is the Euclidean algorithm for successive divisions. Following this, the error locator polynomial $\sigma(x)$ is determined, enabling the identification of error positions. Subsequently, the error evaluator polynomial $Z_0(x)$ is computed, allowing to deduce error values at specific positions. The formula for calculating the error value at position j_i is the following:

$$e_{j_i} = \frac{-Z_0(\alpha^{-j_i})}{\sigma'(\alpha^{-j_i})}$$

where $\sigma'(x)$ is the first derivative of $\sigma(x)$ with respect to x.

The final step computes the difference between the received polynomial and the estimated error polynomial to derive the estimated transmitted codeword.

3.4 Reed Muller codes

Reed-Muller (RM) codes are linear block codes that can be defined in terms of Boolean functions [10]. Specifically, to define codes of length $n = 2^m$, we need m variables, namely $v_1, v_2, \ldots, v_m$, which take values in $\mathbb{F}_2$. Let $\mathbf{v} = (v_1, \ldots, v_m)$ range over $\mathbb{F}_2^m$, the set of all 2^m binary m-tuples.

v_3	v_2	v_1	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 3.1: Example of a truth table for $m = 3$

Definition 3.3. *A function $f(\mathbf{v}) = f(v_1, \cdots, v_m)$ that takes on the values 0 and 1 is called a Boolean function.*

Such a function can be specified by a truth table, which is a list of all possible input combinations along with their corresponding outputs. For example, when $m = 3$, a possible Boolean function is the one specified by Table 3.1. The last column of the truth table is a binary vector of length $n = 2^m$, denoted by $\mathbf{f}$, obtained from the Boolean function $f(v_1 \cdots, v_m)$. The last column can be filled arbitrarily; therefore, there are 2^{2^m} Boolean functions of m variables. Any Boolean function can be expressed as a sum of 2^m functions, namely:

$$1, v_1, v_2, \cdots v_m, v_1v_2, v_1v_3, \cdots v_{m-1}v_m, \cdots, v_1v_2v_3 \cdot \cdots \cdot v_m \tag{3.1}$$

Since there are 2^{2^m} different Boolean functions in total, all these sums must be distinct. In other words, the 2^m vectors corresponding to the functions in equation (3.1) are linearly independent. This property is of paramount importance when constructing Reed-Muller codes, as it ensures that the codewords generated by these codes are distinct and can be reliably decoded.

Definition 3.4. *(r-th binary Reed-Muller code) The r-th binary Reed-Muller code, denoted by $\mathcal{R}(r, m)$, of length $n = 2^m$, for $0 \leqslant r \leqslant m$, is the set of all vectors $\mathbf{f}$, where $f(v_1, \cdots v_m)$ is a Boolean function representable as a polynomial of maximum degree r.*

Since each Boolean function can be expressed as a sum of the functions in (3.1), and since each codeword can be expressed as a Boolean function, as stated in Definition 3.4, it follows that the r-th order RM code consists of all linear combinations of the vectors:

$$1, v_1, \cdots, v_m, v_1v_2, \cdots, v_{m-1}v_m \cdots \textit{(up to degree r)}.$$

In particular, within the context of a $\mathcal{R}(r, m)$ code, the code consists of (the vector corresponding to) all polynomials in the binary variables $v_1 \cdots, v_m$ of degree equal to or less than r.

Theorem 3.2. *For any positive integers m and r where $0 \leqslant r \leqslant m$, there exists a binary r-th order Reed-Muller code denoted by $\mathcal{R}(r, m)$ with the following parameters:*

- *code length:* $n = 2^m$
- *dimension:* $k = \sum_{i=0}^{r} \binom{m}{i}$
- *Minimum distance:* $d = 2^{m-r}$

For example, let's consider the code denoted by $\mathcal{R}(1, 2)$. This is a Reed Muller code with $r = 1$ and $m = 2$; therefore, $n = 2^m = 4$ and $k = m + 1 = 3$. This code consists of $2^3 = 8$ codewords each of length n. Specifically, a generic codeword can be expressed as a linear combination of k basis functions, namely:

$$a_0\mathbf{1} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2$$

where $\mathbf{1} = (1, 1, 1, 1)$, $v_2 = (0, 0, 1, 1)$ and $v_1 = (0, 1, 0, 1)$ form a basis of the RM code and a_i for $i = 0, \cdots k - 1$ is a scalar in $\mathbb{F}_2$. In this way we obtain the codewords in Table 3.2. For instance, the codeword associated with the information sequence 001 of length k is 0011 and so on for the others.

As we proceed with the development of this thesis, our attention will be directed towards the exploration of first-order Reed-Muller codes. Therefore,

a_0	a_1	a_2	Codeword
0	0	0	0000
0	0	1	v_2 =0011
0	1	0	v_1 =0101
0	1	1	$v_1 + v_2$ =0110
1	0	0	1111
1	0	1	1111+v_2=1111+0011=1100
1	1	0	1111+v_1=1111+1100=0011
1	1	1	1111+v_1 + v_2=1111+0011+0101=1001

Table 3.2: Codeword for a $\mathcal{R}(1, 2)$ code

the subsequent discussion will delve into an in-depth examination and analysis of these codes, aiming to provide a comprehensive understanding of their properties and features.

3.4.1 First order Reed Muller codes

A first-order Reed-Muller code can be considered as a special case of RM codes described in section 3.4 when $r = 1$. For any vector $\mathbf{u} = (u_1, \ldots, u_m) \in \mathbb{F}_2^m$, let $f(\mathbf{u})$ represent the value of f at $\mathbf{u}$, or equivalently, the component of $\mathbf{f}$ corresponding to $\mathbf{u}$. It will be useful to define $\mathbf{F}$ as the real vector obtained from the binary vector $\mathbf{f}$ by replacing 1's with -1's and 0's with +1's. Thus, the component of $\mathbf{F}$ at the position corresponding to $\mathbf{u}$ is:

$$F(\mathbf{u}) = (-1)^{f(\mathbf{u})} \tag{3.2}$$

Before discussing first-order Reed-Muller codes, let's revisit some fundamental concepts related to Hadamard matrices [11].

Definition 3.5. *(Hadamard Matrix) A Hadamard matrix* $\mathbf{H}$ *of order* n *is a* $n \times n$ *matrix of +1's and -1's such that* $\mathbf{H}\mathbf{H}^T = n\mathbf{I}_n$ *where* $\mathbf{I}_n$ *is the identity matrix of order* n*.*

In accordance with the previous definition, the real inner product of any two distinct rows of $\mathbf{H}$ is zero, therefore, distinct rows are orthogonal. Additionally,

the real inner product of any row with itself is n. On top of that, since we have $\mathbf{H}\mathbf{H}^{\mathrm{T}} = n\mathbf{I}_n \leftrightarrow \mathbf{H}^{\mathrm{T}}\mathbf{H} = n\mathbf{I}_n$, the columns have the same properties. There are several methods for constructing Hadamard matrices, one of the most commonly used is the Sylvester construction. In accordance with this, starting from the definition of the 1st-order Hadamard matrix, which is $\mathbf{H_1} = (1)$, the construction proceeds recursively. In particular, the Hadamard matrix of order $2n$, with n being any power of 2, is defined as follows:

$$\mathbf{H}_{2n} = \begin{pmatrix} \mathbf{H}_n & \mathbf{H}_n \\ \mathbf{H}_n & -\mathbf{H}_n \end{pmatrix}.$$

In particular, it's important to highlight that Hadamard matrices constructed using the Sylvester procedure are symmetric, thus, it follows that $\mathbf{H} = \mathbf{H}^{\mathrm{T}}$, hence, $\mathbf{H}\mathbf{H}^{\mathrm{T}} = \mathbf{H}\mathbf{H} = n\mathbf{I}_n$.

Let $\mathbf{H}$ be a Hadamard matrix of order n. Assuming $\mathbf{F}$ is a real vector of length 2^m, its Hadamard transform (or Walsh transform) is a vector of length 2^m defined by:

$$\hat{\mathbf{F}} = \mathbf{F} \cdot \mathbf{H} \tag{3.3}$$

The entries $(-1)^{\mathbf{u}\cdot\mathbf{v}}$ for $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^m$ form a Hadamard matrix of order $n = 2^m$. Therefore, $\mathbf{H}$ is a $2^m \times 2^m$ Hadamard matrix given by $\mathbf{H} = (H_{u,v})$ where $H_{u,v} = (-1)^{\mathbf{u}\cdot\mathbf{v}}$ with $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^m$. From (3.3), it follows that:

$$\hat{F}(\mathbf{u}) = \sum_{\mathbf{v}\in\mathbb{F}_2^m} (-1)^{\mathbf{u}\times\mathbf{v}} \cdot F(\mathbf{v}), \quad \mathbf{u} \in \mathbb{F}_2^m. \tag{3.4}$$

From (3.2), by substituting into (3.4), we obtain:

$$\hat{F}(\mathbf{u}) = \sum_{\mathbf{v}\in\mathbb{F}_2^m} (-1)^{\mathbf{u}\cdot\mathbf{v}} \cdot (-1)^{f(\mathbf{v})} = \sum_{\mathbf{v}\in\mathbb{F}_2^m} (-1)^{\mathbf{u}\cdot\mathbf{v}+f(\mathbf{v})} \tag{3.5}$$

According to (3.5), $\hat{F}(\mathbf{u})$ is equal to the difference between the number of 0's and the number of 1's in the binary vector $\mathbf{u} \cdot \mathbf{v} + \mathbf{f}$. This is a vector of

length 2^m obtained by iterating over $\mathbf{v} \in \mathbb{F}_2^m$. Specifically, note that when $\mathbf{u} \cdot \mathbf{v} + f(\mathbf{v}) = \sum_{i=1}^{m} u_i \cdot v_i + f(\mathbf{v}) = 0$, the term $(-1)^{\mathbf{u}\cdot\mathbf{v}+f(\mathbf{v})}$ is equal to +1. On the other hand, when $\mathbf{u} \cdot \mathbf{v} + f(\mathbf{v}) = \sum_{i=1}^{m} u_i \cdot v_i + f(\mathbf{v}) = 1$, the term $(-1)^{\mathbf{u}\cdot\mathbf{v}+f(\mathbf{v})}$ is equal to -1. Based on this:

$$\hat{F}(\mathbf{u}) = 2^m - 2 \cdot d\left(f, \quad \sum_{i=1}^{m} u_i \cdot v_i\right), \quad \mathbf{v} \in \mathbb{F}_2^m \tag{3.6}$$

where $d(\cdot)$ is a metric function.

Clearly, $\sum_{i=1}^{m} u_i \cdot v_i$ is an inner product, making it a scalar in $\mathbb{F}_2$. However, according to (3.6), we need to iterate over $\mathbf{v} \in \mathbb{F}_2^m$. By doing so, we obtain a vector with 2^m components since there are 2^m distinct possible $\mathbf{v} \in \mathbb{F}_2^m$. This approach allows us to compute the distance with respect to $\mathbf{f}$, which is also a vector with 2^m components. By rearranging the terms in (3.6), we derive the following equations:

$$d\left(f, \quad \sum_{i=1}^{m} u_i \cdot v_i\right) = \frac{1}{2} \cdot \left(2^m - \hat{F}(\mathbf{u})\right) \tag{3.7}$$

$$d\left(f, \quad 1 + \sum_{i=1}^{m} u_i \cdot v_i\right) = \frac{1}{2} \cdot \left(2^m + \hat{F}(\mathbf{u})\right) \tag{3.8}$$

where $\sum u_i v_i$ represents every possible codeword of the RM code.

Therefore, equations (3.7) and (3.8) show that the closest codeword to $\mathbf{f}$ is the one for which $\left|\hat{F}(\mathbf{u})\right|$ is maximum [12]. This last consideration forms the basis of the decoding procedure for Reed-Muller codes.

3.4.2 Encoding and Decoding $\mathcal{R}(1, m)$

The first-order Reed-Muller code, denoted as $\mathcal{R}(1, m)$, is a $[2^m, m + 1, 2^{m-1}]$ linear block code. When it comes to encoding, the procedure is straightforward. Since RM codes are linear block codes, encoding is performed using a generator matrix, following the same procedure described in subsection 2.1.1.

The generator matrix for the $\mathcal{R}(1,m)$ code comprises $m+1$ rows and 2^m columns. The first row is consistently filled with 2^m 1s. Subsequently, the matrix is constructed column by column (excluding the first row), utilizing the binary representation of decimal numbers ranging from 0 to 2^m-1. For illustrative purposes, let's consider the $\mathcal{R}(1,3)$ code. The generator matrix has dimensions of $k \times n$, specifically 4×8. The initial row is filled with 1s, and then, column by column, we observe the binary representation of decimal numbers from 0 to 7. Specifically:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Based on the structure of the generator matrix of a $\mathcal{R}(1,m)$ code, it can be said it is the dual code of the Hamming code.

When it comes to decoding, Maximum Likelihood Decoding (MLD) requires comparing the received vector $\mathbf{f}$ with every codeword of the code. Based on this, we need to find the distance from $\mathbf{f}$ to every codeword of $\mathcal{R}(1,m)$ and then decode $\mathbf{f}$ as the closest codeword. Referring to equations (3.7) and (3.8), this is equivalent to finding the largest component of $\left|\hat{F}(\mathbf{u})\right|$. For a better understanding, it is important to note that a generic codeword of a Reed Muller code can be expressed as a linear combination of $\mathbf{1}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \cdots, \mathbf{v}_m$. Therefore, we have $m+1$ base functions. Specifically, $\sum_{i=1}^{m} u_i v_i$ represents a linear combination of these base functions with weights given by the information sequence $u_0, u_1, u_2, \cdots, u_m$. A generic codeword $c \in \mathcal{R}(1,m)$ can be expressed as:

$$\mathbf{c} = u_0 \cdot \mathbf{1} + u_1 \cdot \mathbf{v}_1 + u_2 \cdot \mathbf{v}_2 + \cdots + u_m \cdot \mathbf{v}_m$$

Assuming the largest component of $\left|\hat{F}(\mathbf{u})\right|$ is $\left|\hat{F}(u_1, u_2, \cdots u_m)\right|$, we have to

distinguish two different cases according to its sign, specifically:

- $\hat{F}(u_1, u_2, \cdots u_m) \geqslant 0 \rightarrow$ from (3.7) we decode $\mathbf{f}$ as $\sum_{i=1}^{m} u_i v_i$
- $\hat{F}(u_1, u_2, \cdots u_m) < 0 \rightarrow$ from (3.8) we decode $\mathbf{f}$ as $1 + \sum_{i=1}^{m} u_i v_i$

It is important to notice that Hadamard transform (also known as Walsh-Hadamard Transform and denoted by WHT) requires the construction of the Hadamard matrix, therefore, especially when $n = 2^m$ is large this could lead to a noticeable memory occupation. On top of that, Hadamard transform requires $O(2^{2m})$ operations since for direct calculation of $\hat{\mathbf{F}} = \mathbf{F} \cdot \mathbf{H}_{2^m}$ are needed 2^m products and $2^m - 1$ additions for each of the 2^m components of $\mathbf{F}$. Therefore, overall $2^m \cdot 2^m = 2^{2m}$ products and $2^m \cdot (2^m - 1)$ additions, so, the total number of required operations is $2^{2m} + 2^m \cdot (2^m - 1)$ which is of the order of $O(2^{2m})$. Fortunately, there is a faster way to obtain $\hat{\mathbf{F}}$ which is called Fast Walsh-Hadamard Transform (FWHT).

3.4.3 Fast Walsh-Hadamard Transform (FWHT)

The Fast Walsh-Hadamard Transform (FWHT) is a powerful and efficient algorithm designed specifically for computing the Walsh-Hadamard Transform. A direct WHT implementation of order $n = 2^m$ requires $O(2^{2m})$ computations. In contrast, the FWHT employs a different approach, reducing the complexity to $2^m \cdot \log_2(2^m)$ additions and subtractions, offering a significant performance enhancement. The success of the FWHT derives from its divide-and-conquer strategy, recursively decomposing a WHT of size n into two smaller WHTs of size $n/2$. This approach is rooted in the recursive definition of the $2^m \times 2^m$ Hadamard matrix $\mathbf{H}_{2^m}$ and offers a systematic and efficient way to compute the WHT.

To further illustrate the advantages of the FWHT, let's consider a straightforward example with $m = 3$, resulting in $n = 2^3 = 8$. This basic example can serve as a stepping stone for understanding the principles and benefits of the

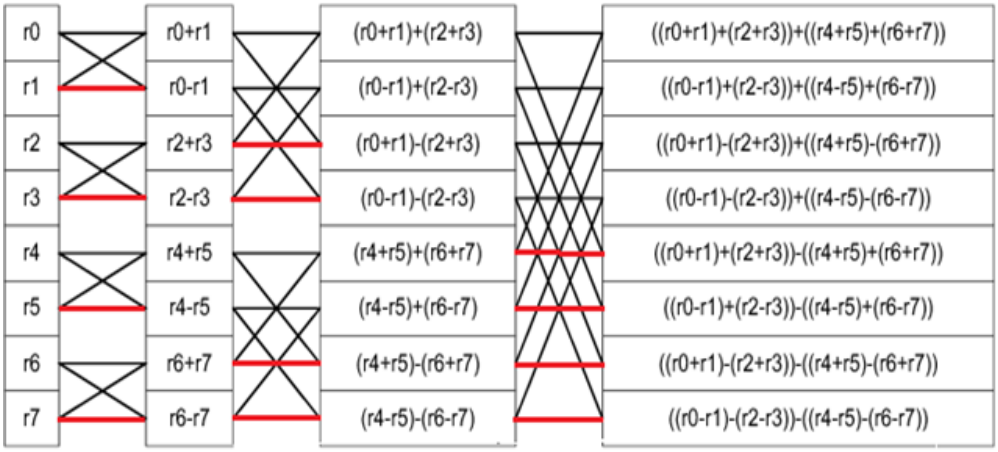

Figure 3.1: FWHT structure for $m = 3$

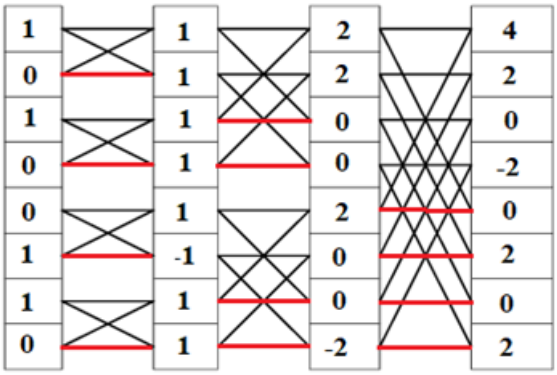

Figure 3.2: FWHT example assuming $\mathbf{r} = (r_0, r_1, \cdots r_7) = (10100110)$

FWHT in more complex applications and scenarios. The implementation of FWHT relies on the structure illustrated in Figure 3.1. For instance, assuming $\mathbf{r} = (r_0, r_1, \cdots r_7) = (10100110)$, we obtain the results shown in Figure 3.2. First, it is important to note that both the input and output are sequences of length n. Secondly, it's important to recognize that the number of stages is equal to $\log_2 n = 3$, as is typical in a divide-and-conquer algorithm. At each stage, $n = 2^m$ additions (indicated by the black branches) or subtractions (indicated by the red branches) must be calculated. Therefore, based on the above observations, the overall number of operations is in the order of $O(n \log_2 n)$. This demonstrates how an implementation that relies on the structure described in Figure 3.1 offers significant computational advantages.

3.4.4 Encoding a $\mathcal{R}(1, m)$ code through FWHT

As described in subsection 3.4.2, encoding a first-order Reed-Muller code follows the standard approach of multiplying the information sequence by the generator matrix $\mathbf{G}$ of the code. However, alongside this encoding method, there exists another efficient and fully equivalent approach that uses the Fast Walsh-Hadamard Transform described in subsection 3.4.3. In this subsection, we will provide a detailed explanation of the encoding method based on the FWHT, and we will further clarify its equivalence with the classical approach.

The encoding process for a first-order Reed-Muller code, using the Fast Walsh-Hadamard Transform, begins by specifying a vector $\mathbf{Y}$. Just like in encoding, we express the information sequence as an integer modulo 2^k denoted as $a \in \mathbb{Z}_{2^k}$. Then, we define $\mathbf{Y}$ such that it consists of a single non-zero element, positioned at index a when $a < n$, or at index $a - n$ when $a \geqslant n$. In this configuration, the only non-zero value in $\mathbf{Y}$ is n when $a < n$ or $-n$ otherwise. Specifically:

$$\mathbf{Y} = [0 \ \cdots 0 \ +n \ 0 \ \cdots 0] \quad \text{where } n \text{ is indexed by } a < n$$

$$\mathbf{Y} = [0 \ \cdots 0 \ -n \ 0 \ \cdots 0] \quad \text{where } -n \text{ is indexed by } a - n \text{ when } a \geqslant n$$

Let $\mathbf{y}$ be the output of the FWHT applied to $\mathbf{Y}$. Specifically, as $\mathbf{Y}$ contains a single non-zero term, calculating its FWHT means considering only one row of the Hadamard matrix of order $n = 2^m$ multiplied by a scale factor of n. Specifically, the row indexed by a if $a < n$, otherwise, the row indexed by $a - n$ of the flipped Hadamard matrix multiplied by n. For simplicity, let's imagine constructing a matrix $\mathbf{H} \in \mathbb{F}_{\{\pm 1\}}^{2n \times n}$ in the following manner:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H_n} \\ -\mathbf{H_n} \end{pmatrix}$$

where $\mathbf{H_n}$ is a Hadamard matrix of order $n = 2^m$.

The vector $\mathbf{y}$ can be seen as the row of $\mathbf{H}$ index by a and multiplied by n. Denoting as $\mathbf{h}_a$ the row of $\mathbf{H}$ indexed by a, we obtain that:

$$\mathbf{y} = n\mathbf{h}_a$$

Based on what has been said, it follows that $\mathbf{y}$ is in $\mathbb{F}^n_{\{\pm n\}}$. At this point, the vector $\mathbf{y}$ is turned into codeword using the mapping $\{+n \to 0, -n \to +1\}$. After having seen how the encoding process using the FWHT works, we want to verify the equivalence with respect to the classic encoding approach which involves matrix multiplication between the information vector of size $1 \times k$ where $k = m + 1$ and the generator matrix $\mathbf{G}$. First, given a as an integer number between 0 and $2^k - 1$, the vector representation is considered. For example, if $m = 2$, $k = 3$ and $a = 3$, the vector representation of a denoted by $\mathbf{a}$ is [0 1 1]. In this case, the codeword is simply given by the sum of the row at index 1 and the row at index 2 of $\mathbf{G}$ consistently with the position of the 1's in $\mathbf{a}$. In particular:

$$codeword = (011) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} = [0110] \tag{3.9}$$

At this point, to understand the equivalence between the two encoding approaches, it is first necessary to consider the link between the $\mathbf{G}$ and the Hadamard matrix of order $n = 2^m$. Returning to the example, the Hadamard matrix of order $n = 2^m = 4$ is such that the row at the index given by a, is exactly the sum of the row at index 1 and the row at index 2 of $\mathbf{G}$ considering the usual mapping. According to this, the encoding by matrix multiplication is equivalent to the encoding procedure which exploits the FWHT. The

Hadamard matrix of order $n = 4$ is:

$$\mathbf{H}_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

The row at index $a = 3$ of $\mathbf{H}_4$ is $(1\ -1\ -1\quad 1) \rightarrow (0\,1\,1\,0)$; therefore, the codeword obtained is equal to the one in (3.9). Now, for completeness we show another example where it turns out that $a > n$. For example, let's keep the same set of parameters but suppose $a = 5$, hence $\mathbf{a}$=(1 0 1). In this case, the codeword is simply given by the sum of the row at index 0 and the row at index 2 of $\mathbf{G}$ consistently with the position of the 1's in $\mathbf{a}$. In particular:

$$codeword = (101) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} = [1010] \tag{3.10}$$

The Hadamard matrix of order $n = 4$ is such that the row at the index given by $a-n$, therefore index 1, is the sum of the row with index 0 and the row with index 2 of $\mathbf{G}$ properly flipped and considering the usual mapping. Indeed, the row at index 1 of $\mathbf{H}_4$ is $(1\ -1\quad 1\ -1) \rightarrow (0\,1\,0\,1)$ which is exactly the flipped version of the codeword in (3.10).

3.4.5 Decoding a $\mathcal{R}(1, m)$ code through FWHT

Let $\mathbf{r} \in \mathbb{F}_2^n$ be the received vector, hence, $\mathbf{r} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathbb{F}_2^n$ denotes the transmitted codeword and $\mathbf{e} \in \mathbb{F}_2^n$ the error pattern. First of all, the received sequence $\mathbf{r}$ undergoes a transformation into soft values using the usual mapping $\{0 \rightarrow +1, 1 \rightarrow -1\}$, which is the reverse mapping compared to the one employed in the encoding process described in the previous subsection.

The resulting soft values version of $\mathbf{r}$ is denoted as $\mathbf{z} \in \mathbb{F}^n_{\{\pm 1\}}$. Initially, we refer to the ideal case for which the error pattern $\mathbf{e}$ is the null vector and $\mathbf{r} = \mathbf{c}$. Thus, the soft version of $\mathbf{r}$ is exactly the vector $\mathbf{y}$ defined during encoding but divided by the scaling factor n, in symbols, it can be expressed as:

$$\mathbf{z} = \frac{\mathbf{y}}{n} \tag{3.11}$$

At this point, the next step of the decoding procedure involves computing the vector $\mathbf{Z}$ as the FWHT of $\mathbf{z}$, formally:

$$\mathbf{Z} = \text{FWHT}(\mathbf{z}, n) = \mathbf{z} \cdot \mathbf{H}_n \tag{3.12}$$

Substituting equation (3.11) into equation (3.12), we obtain:

$$\mathbf{Z} = \frac{\mathbf{y}}{n} \cdot \mathbf{H}_n \tag{3.13}$$

Given that $\mathbf{y} = \text{FWHT}(\mathbf{Y}, n) = \mathbf{Y} \cdot \mathbf{H}_n$, by substituting in equation (3.13), we obtain:

$$\mathbf{Z} = \frac{\mathbf{Y} \cdot \mathbf{H}_n}{n} \cdot \mathbf{H}_n \tag{3.14}$$

Since, $\mathbf{H}_n \cdot \mathbf{H}_n = n \cdot \mathbf{H}_n$, it follows that:

$$\mathbf{Z} = n\mathbf{I}_n \cdot \mathbf{Y} = \mathbf{Y} \tag{3.15}$$

Therefore, we obtain exactly the vector $\mathbf{Y}$ as defined in the encoding process. With $\mathbf{Y}$ at hand, the recovery of the encoded information represented as an integer $a \in \mathbb{Z}_{2^k}$ becomes a straightforward task. Initially, in accordance with equations (3.7) and (3.8), we determine the maximum absolute value within $\mathbf{Y}$ and store the index i at which it is maximum. Subsequently, if $\mathbf{Y}(i)$ is greater than 0, it implies that $a = i$, while, if $\mathbf{Y}(i) \leqslant 0$, it means that $a = i + n$. This process allows us to accurately recover the original encoded information.

On top of that, in this context, it is extremely important to note that calculating the FWHT($\mathbf{z}$,n) is computationally equivalent to summing the n rows of the Hadamard matrix of order n, taken as they are if the corresponding element of $\mathbf{z}$ is 1, or flipped (multiplied by -1) otherwise. Clearly, in the ideal case where the error pattern $\mathbf{e} \in \mathbb{F}_2^n$ is the zero vector, following this procedure exactly results in $\mathbf{Z} = \text{FWHT}(\mathbf{z}, n) = \mathbf{Y} = [0 \ \cdots 0 \ \pm n \ 0 \ \cdots 0]$.

Up to this point, we have delved into the ideal scenario, where the error pattern is represented by the zero vector. Nevertheless, in the general case, it's essential to acknowledge that $\mathbf{e}$ is not the zero vector; hence, $\mathbf{r}$ differs from $\mathbf{c}$. This implies that the soft version of $\mathbf{r}$, formerly denoted as $\mathbf{z}$, can no longer be determined using the equation (3.11). Instead, it becomes imperative to account for the presence of the error pattern. In the aftermath of performing the Fast Walsh-Hadamard Transform (FWHT), we obtain a vector with dimensions of $1 \times n$, where its constituents are no longer exclusively zero except for one element equal to $\pm n$. Instead, the resulting vector is typically "contaminated" by the influence of the error pattern. The significance of this deviation becomes most apparent as the Hamming weight of $\mathbf{e}$ increases; as it does, the values within the resulting vector $\mathbf{Z}$ after the application of the FWHT($\mathbf{z}, n$) exhibit a tendency to "flatten out". In more explicit terms, there is no longer a single dominant value that stands out. Naturally, as the Hamming weight of the error pattern grows excessively high, it is entirely possible for the maximum value within the vector $\mathbf{Z}$ not to correspond to the position a (for $a < n$) or the position $a - n$ (for $a \geqslant n$), leading to a decoding failure.

3.5 Concatenated codes

Concatenated codes are error-correcting codes that are constructed from two or more simpler codes to achieve good performance with reasonable complexity. A graphic representation of a concatenated code is shown in Figure 3.3.

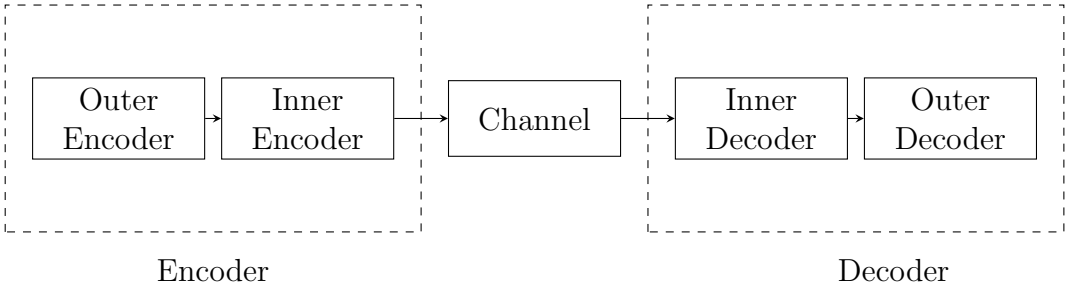

Figure 3.3: Illustrative representation of a concatenated code constructed using an inner code and an outer code.

Going into more mathematical detail, let $\mathbb{F}_2^m$ be an extension field of $\mathbb{F}_2$. Each element in $\mathbb{F}_2^m$ can be represented by an m-tuple over $\mathbb{F}_2$.

Definition 3.6. *(Concatenated codes) A concatenated code comprises a non-binary external code $\mathcal{C}_1$ described by the triplet $[n_1, k_1, d_1]$ over $\mathbb{F}_2^m$, along with an internal binary code $\mathcal{C}_2$ described by the triplet $[n_2, k_2, d_2]$ where $k_2 = m$.*

Let $\mathbf{u} = (u_0, \cdots u_{n_1-1}) \in \mathbb{F}_2^{n_1}$ be a codeword in $\mathcal{C}_1$. By expanding each code symbol of $\mathbf{u}$ into an m-tuple over $\mathbb{F}_2$, we obtain an $m \cdot n_1$-tuple $\mathbf{w} = (w_0, \cdots w_{m \cdot n_1 - 1})$ over $\mathbb{F}_2$. Each set of m consecutive binary symbols in $\mathbf{w}$ is encoded into a codeword in $\mathcal{C}_2$. This process results in a sequence of $n_1 n_2$ binary symbols, consisting of a sequence of n_1 binary codewords in $\mathcal{C}_2$. This concatenated sequence of $n_1 n_2$ binary symbols contains $k_1 m$ information bits. Since there are $2^{k_1 m}$ codewords in $\mathcal{C}_1$, there exist $2^{k_1 m}$ such concatenated $n_1 n_2$-bit sequences. These sequences together form a binary $(n_1 n_2, k_1 m)$ linear block code known as a concatenated code.

The encoding of a concatenated code consists of two stages. In the first stage, a binary information sequence of $k_1 m$ bits is divided into k_1 bytes, each containing m information bits. Each m-bit byte is considered an element in $\mathbb{F}_2^m$. During the first stage of encoding, the k_1 bytes, treated as k_1 information symbols over $\mathbb{F}_2^m$, are encoded into an n_1-byte codeword $\mathbf{u}$ in $\mathcal{C}_1$. This first encoding stage produces a coded sequence $\mathbf{w}$ consisting of mn_1 bits (or n_1 m-bit bytes). In the second stage of encoding, each group of m consecutive bits in $\mathbf{w}$ is encoded into an n_2-bit codeword in $\mathcal{C}_2$, resulting in a string of n_1

codewords in $\mathcal{C}_2$. This string of n_1 codewords in $\mathcal{C}_2$ is then transmitted one C_2 codeword at a time, sequentially. Given that $\mathcal{C}_1$ and $\mathcal{C}_2$ are used in the outer and inner encoding stages, respectively, they are referred to as the outer and inner codes, respectively. If the minimum distances of $\mathcal{C}_1$ and $\mathcal{C}_2$ are d_1 and d_2, then the minimum distance of the concatenation of $\mathcal{C}_1$ and $\mathcal{C}_2$ is $d_1 d_2$.

The decoding of a concatenated code is also performed in two stages. Initially, decoding is executed for each inner n_2-bit received word as it arrives, based on a decoding method for the inner code $\mathcal{C}_2$. The parity-check bits are then removed, resulting in a sequence of n_1 m-bit bytes. This stage of decoding is known as the inner decoding. The n_1 decoded bytes at the end of the inner decoding are subsequently decoded based on the outer code $\mathcal{C}_1$ using a specific decoding method. This second decoding stage, referred to as the outer decoding, yields k_1 decoded information bytes ($k_1 m$ decoded information bits).

3.5.1 Tensor product codes

Tensor product codes can be seen as an example of concatenated codes by considering their construction and how they incorporate the principles of concatenated coding schemes. Concatenated codes are formed by applying two or more error-correcting codes in sequence, where the output of one code becomes the input to the next. Tensor product codes follow a similar concept but use a mathematical operation called the tensor product to achieve this.

Definition 3.7. *Given two codes $\mathcal{C}_1$ and $\mathcal{C}_2$, with dimensions k_1 and k_2 and lengths n_1 and n_2 such that $k_2 = n_1$, we denote by $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$ their tensor product.*

Namely, the codewords of $\mathcal{C}$ can be interpreted as all the $n_2 \times n_1$ matrices whose rows are codewords of $\mathcal{C}_1$ and columns are codewords of $\mathcal{C}_2$.

Chapter 4

Security Background

Public key cryptosystems use mathematical functions with trapdoor properties to ensure that information encrypted with a public key can only be decrypted efficiently with the corresponding private key. Moreover, the security of the system relies on the computational infeasibility of deriving the private key from the public key. In this regard, before delving into the concept of security for a public key cryptosystem, it is necessary to first define what is meant by trapdoor and one-way function.

Definition 4.1. *(Trapdoor function) A trapdoor is a mathematical function that is straightforward to compute in one direction but extremely challenging to reverse, requiring access to a secret piece of information, usually referred to as the "trapdoor".*

Definition 4.2. *(One-way function) A one-way function is a mathematical function that is relatively easy to compute in one direction (i.e., given an input, it's easy to compute the output) but computationally difficult to reverse (i.e., given the output, it is extremely hard to compute the original input).*

To gain a more intuitive understanding of the functioning of a trapdoor and a one-way function from a graphical perspective, refer to Figures 4.1 and 4.2, respectively.

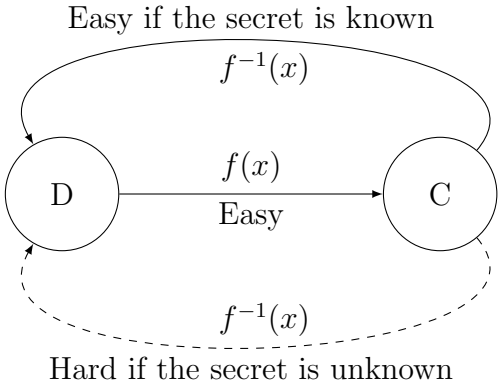

Figure 4.1: Graphical representation of a trapdoor function.

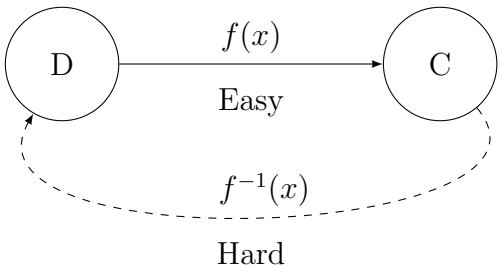

Figure 4.2: Graphical representation of a one-way function.

A trapdoor is used as an encryption function; indeed, in a public key cryptosystem, efficient decryption is only possible when a specific secret is known. In this case, the secret is the private key corresponding to the public key used during encryption. Furthermore, the public key must be inseparably linked to the private key. Therefore, it is necessary to use a one-way function to derive the public key from the secret key. Consequently, the security of a public key cryptosystem is based on the computational difficulty of solving a mathematical problem. This problem can involve the inversion of a trapdoor function without knowledge of the secret. Alternatively, it may relate to the inversion of a one-way function. In other words, in the first case, an attacker, starting from the encrypted message, tries to recover the corresponding plaintext. In the second scenario, an attacker, starting from the public key, attempts to recover the private key. Clearly, once the secret key is compromised, the attacker can recover all the plaintext encrypted with the corresponding public key.

For instance, consider RSA (Rivest Shamir Adleman), one of the oldest and most widely used public-key cryptosystems in the pre-quantum computing era. RSA's security relies on the challenge of factoring large integers (often hundreds of decimal digits) [13]. Specifically, when given an integer n derived from the product of two carefully selected prime numbers, p and q, it becomes computationally infeasible on classical computers to factorize n. In this case, p and q form part of the private key, while n is included in the public key.

Based on what as been said so far, an attacker aiming to compromise a public-key cryptosystem must solve the underlying mathematical problem. Since there may exist various methods to tackle this problem, the attacker will opt for the most efficient algorithm, characterized by the lowest cost, denoted as 2^{λ_1}. However, it's important to note that the security of the cryptosystem is also influenced by the specific protocol that makes use of this mathematical problem. For instance, an attacker could attempt to exploit the protocol's structure without necessarily needing to discover a valid private key. In such a scenario, assuming the attacker can somehow leverage the protocol's structure within which the cryptosystem is employed, we denote the cost as 2^{λ_2}. The security level of the cryptographic scheme is then determined by the less costly approach between the two, given by:

$$\lambda = \log_2 \min(2^{\lambda_1}, 2^{\lambda_2})$$

In general, when assessing the security of a cryptographic scheme, it is essential to consider two distinct attack approaches: structural attack and non-structural attack. The former involves exploiting the well-understood algebraic structure within the scheme, while the latter focuses on recovering the message or secret key without depending on the algebraic structure, remaining completely agnostic to it.

4.1 Security for code-based cryptosystems

In this section we describe difficult problems which can be used in code-based cryptosystems with a focus on the cryptographic problems at the core of HQC, the post-quantum public key cryptosystem under consideration in this thesis. Traditional code-based cryptography relies on the complexity of decoding a randomly generated linear block code ([14], [15]). Specifically, in the case of a linear block code defined by a generator matrix $\mathbf{G}$, the decoding problem can be formalized as follows:

Problem 1. ***Computational Decoding Problem (C-DP)***
For positive integers n, k and w, on input $(\mathbf{G}, \mathbf{x}) \in \mathbb{F}_2^{k\times n} \times \mathbb{F}_2^n$, the computational decoding problem asks to find $\mathbf{u} \in \mathbb{F}_2^k$ and $\mathbf{e} \in \mathbb{F}_2^n$ such that $\mathbf{x} = \mathbf{u} \cdot \mathbf{G} + \mathbf{e}$, with $wt(\mathbf{e}) = w$.

The problem at hand can be reframed using the parity-check matrix $\mathbf{H}$ of a linear block code, as this matrix uniquely represents such a code (as described in subsection 2.1.1). Additionally, for a clearer exposition of the problem, it is helpful to begin by defining the following:

Definition 4.3. *(SD Distribution) For positive integers n, k and w, the SD Distribution, denoted as SD(n,k,w), chooses $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{x} \xleftarrow{\$} \mathbb{F}_2^n$ such that $wt(\mathbf{x}) = w$, and outputs $(\mathbf{H}, \sigma(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}^T)$.*

The SD distribution produces both the syndrome of $\mathbf{x}$, denoted as $\sigma(\mathbf{x})$, and the parity-check matrix $\mathbf{H}$. It randomly selects $\mathbf{x}$ from $\mathbb{F}_2^n$ and $\mathbf{H}$ from $\mathbb{F}_2^{(n-k)\times n}$. Then, by assuming that the linear block code is represented by a randomly chosen $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, the syndrome of $\mathbf{x}$ is effectively $\sigma(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}^T$. Defining the SD distribution is essential for establishing the subsequent problem.

Problem 2. ***Computational Syndrome Decoding Problem (C-SDP)***
For positive integers n, k and w, on input $(\mathbf{H}, \mathbf{y}^T) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$ from the SD distribution, the computational syndrome decoding problem asks to find $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathbf{H}\mathbf{x}^T = \mathbf{y}^T$ (or, equivalently $\mathbf{x}\mathbf{H}^T = \mathbf{y}$) and $wt(\mathbf{x}) = w$.

Since a linear block code can be equivalently represented in terms of a generator matrix and a parity-check matrix, it is reasonable to assume that Problem 1 and Problem 2 are equivalent. This trivial intuition is confirmed in the following theorem:

Theorem 4.1. *C-DP and C-SDP are equivalent in terms of complexity.*

Proof. Let us suppose to receive a challenge for C-DP. On input $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ and $\mathbf{x} \in \mathbb{F}_2^n$, it is possible to compute the parity-check matrix $\mathbf{H}$ corresponding to the generator matrix $\mathbf{G}$ of the linear block code since it is $\mathbf{G} \cdot \mathbf{H}^T = 0$. Then, once the parity-check matrix $\mathbf{H}$ is known, it is possible to compute the syndrome of $\mathbf{x}$, namely:

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{H}^T \tag{4.1}$$

Given that $\mathbf{x} = \mathbf{u} \cdot \mathbf{G} + \mathbf{e}$, by substituting in eq. (4.1), it follows:

$$\mathbf{y} = (\mathbf{u} \cdot \mathbf{G} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \tag{4.2}$$

since $(\mathbf{u} \cdot \mathbf{G}) \cdot \mathbf{H}^T = \mathbf{0}$ because $\mathbf{u} \cdot \mathbf{G}$ is a codeword.
At this stage, by solving C-SDP, it is possible to find $\mathbf{e}$, which can then be used to compute $\mathbf{u} \cdot \mathbf{G} = \mathbf{x} - \mathbf{e}$, and thus $\mathbf{u} \in \mathbb{F}_2^k$, also solving the C-DP. Therefore, C-SDP and C-DP are equivalent when it comes to complexity. □

In light of the established equivalence between C-DP (Computational Decoding Problem) and C-SDP (Computational Syndrome Decoding Problem), we will henceforth refer only to C-SDP. Specifically, rather than the Computational Syndrome Decoding Problem, we will refer to the same problem in its decisional version. This problem is defined below.

Problem 3. ***Decisional Syndrome Decoding Problem (DSD)***
On input $(\mathbf{H}, \mathbf{y}^T) \in \mathbb{F}_2^{(n-k) \times n} \times \mathbb{F}_2^{(n-k)}$*, the decisional syndrome decoding problem asks to decide with non-negligible advantage whether or not* $(\mathbf{H}, \mathbf{y}^T)$ *came from the* $SD(n, k, w)$ *distribution or the uniform distribution over* $\mathbb{F}_2^{(n-k) \times n} \times \mathbb{F}_2^{(n-k)}$.

Generally, a decisional problem is centered around determining the existence of a solution that satisfies certain conditions, while a computational problem focuses on the actual search for such a solution. More specifically, for the computational SDP, the solution is the vector $\mathbf{x} \in \mathbb{F}_2^n$ that satisfies the parity-check equations. On the other hand, for the decisional SDP, the solution is "YES" if $(\mathbf{H}, \mathbf{y}^T)$ came from the $\mathrm{SD}(n, k, w)$ distribution, and therefore, if there exists $\mathbf{x} \in \mathbb{F}_2^n$ such that $\mathbf{y} = \mathbf{x} \cdot \mathbf{H}^T$. Otherwise, the solution is "NO" if such a vector does not exist.

4.2 HQC underlying problems

As Hamming Quasi Cyclic (HQC) employs Systematic Quasi-Cyclic codes (described in subsection 2.2.4), it is necessary to provide an explicit definition of the problems upon which the cryptosystem depends. To specialize the previous discussion to the case of QC codes, it is useful to generalize Definition 4.3 to the case of Systematic Quasi Cyclic codes with index s (Definition 2.9), obtaining the following definition:

Definition 4.4. *(s-QCSD Distribution) For positive integers n, k, w and s, the s-QCSD(n, w) Distribution chooses uniformly at random a parity check matrix $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(sn-n)\times sn}$ of a systematic QC code $\mathcal{C}$ of index s and rate $1/s$ together with a vector $\mathbf{x} = (\mathbf{x}_0, \cdots \mathbf{x}_{s-1}) \xleftarrow{\$} \mathbb{F}_2^{sn}$ such that $wt(x_i) = w$ for $i = 0, \cdots s-1$ and outputs $(\mathbf{H}, \mathbf{H}\mathbf{x}^T) \in \mathbb{F}_2^{(sn-n)\times sn} \times \mathbb{F}_2^{sn-n}$.*

Now, expanding on the coherent line of thought that has been established in the preceding subsection, we progress to articulate the computational formulation of the SDP problem concerning Quasi Cyclic codes. Following this, we will shift our focus towards a comprehensive examination of its decisional counterpart, highlighting certain vulnerabilities that pave the way for the emergence of a distinct improved version.

Problem 4. ***Computation s-QCSD Problem***

For positive integers n, w, and s, consider a randomly chosen parity-check matrix $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(sn-n)\times sn}$ originating from a systematic QC code $\mathcal{C}$ of index s and rate $1/s$, along with a randomly generated vector $\mathbf{y} \xleftarrow{\$} F_2^{sn-n}$ (syndrome). The computational s-Quasi-Cyclic SD Problem entails the task of finding a vector $\mathbf{x} = (\mathbf{x}_0, \cdots \mathbf{x}_{s-1}) \in F_2^{sn}$ such that $w(\mathbf{x}_i) = w$ for every $\mathbf{x}_i \in \mathbb{F}_2^n$ with $i = 0 \cdots s-1$ and such that $\mathbf{y} = \mathbf{x}\mathbf{H}^T$.

As usual, it is advisable to define the problem in its decisional version.

Problem 5. ***Decisional s-QCSD Problem***

On input $(\mathbf{H}, \mathbf{y}^T) \in \mathbb{F}_2^{(sn-n)\times sn} \times \mathbb{F}_2^{sn-n}$, the decisional s-QCSD problem asks to determine with non-negligible advantage whether or not $(\mathbf{H}, \mathbf{y}^T)$ came from the s-QCSD(n, w) distribution or the uniform distribution over $\mathbb{F}_2^{(sn-n)\times sn} \times \mathbb{F}_2^{sn-n}$.

The definition of the Decisional s-QCSD Problem, as presented in Problem 5, brings to light the presence of relatively straightforward distinguishers [16], that inherently pose a challenge to its security. These distinguishers are explicitly designed with the purpose of differentiating a "genuine" syndrome (generated from the s-QCSD distribution) from a completely random syndrome (generated from the uniform distribution). To enhance the comprehension of these distinguishers and their functionality, we will provide preliminary insights into their mechanisms. After a thorough exploration of these distinguishers, we will then describe the strategic solution adopted by HQC's authors to effectively address and mitigate these security concerns. Our approach commences with two fundamental theorems, serving as the cornerstone in our quest to gain a deeper insight into the operations of these distinguishers and their implications. These theorems are of paramount importance in facilitating our understanding of how these distinguishers operate and why they hold such significance in the context of the Decisional s-QCSD Problem.

Theorem 4.2. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be vectors in* $\mathbb{F}_2^n$*. Let* $wt(\mathbf{a})$ *and* $wt(\mathbf{b})$ *be the Hamming weights of* $\mathbf{a}$ *and* $\mathbf{b}$*, respectively. Then it follows that:*

$$wt(\mathbf{a} \cdot \mathbf{b}) = wt(\mathbf{a}) \cdot wt(\mathbf{b}) - 2 \cdot \#cancellations.$$

Consequently:

- $wt(\mathbf{a})$ *is odd and* $wt(\mathbf{b})$ *is odd, then* $wt(\mathbf{a} \cdot \mathbf{b})$ *is odd;*
- $wt(\mathbf{a})$ *is even and* $wt(\mathbf{b})$ *is even, then* $wt(\mathbf{a} \cdot \mathbf{b})$ *is even;*
- $wt(\mathbf{a})$ *is odd and* $wt(\mathbf{b})$ *is even (or vice versa), then* $wt(\mathbf{a} \cdot \mathbf{b})$ *is even.*

Proof. By assuming $\mathbf{c} = \mathbf{a} \cdot \mathbf{b}$, it follows that:

$$\mathbf{c} = \sum_{i \in supp(\mathbf{a})} \sum_{j \in supp(\mathbf{b})} x^{i+j \bmod n}.$$

Therefore, the number of non-zero coefficients in $\mathbf{c}$ is determined by the product of the number of non-zero coefficients in $\mathbf{a}$, i.e., $\mathrm{wt}(\mathbf{a})$, and the number of non-zero coefficients in $\mathbf{b}$, i.e., $\mathrm{wt}(\mathbf{b})$, from which hypothetical cancellations, which occur during the multiplication, must be subtracted. However, since cancellations occur in pairs, it is valid to conclude the theorem's validity. □

Theorem 4.3. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be vectors in* $\mathbb{F}_2^n$*. Let* $wt(\mathbf{a})$ *and* $wt(\mathbf{b})$ *be the Hamming weights of* $\mathbf{a}$ *and* $\mathbf{b}$*, respectively. Then it follows that:*

$$wt(\mathbf{a} + \mathbf{b}) = wt(\mathbf{a}) + wt(\mathbf{b}) - 2 \cdot \#cancellations.$$

Consequently:

- $wt(\mathbf{a})$ *is odd and* $wt(\mathbf{b})$ *is odd, then* $wt(\mathbf{a} + \mathbf{b})$ *is even;*
- $wt(\mathbf{a})$ *is even and* $wt(\mathbf{b})$ *is even, then* $wt(\mathbf{a} + \mathbf{b})$ *is even;*
- $wt(\mathbf{a})$ *is odd and* $wt(\mathbf{b})$ *is even (or vice versa), then* $wt(\mathbf{a} + \mathbf{b})$ *is odd.*

Proof. The proof is analogous to the one provided for Theorem 4.2. □

In HQC, where QC codes are employed, the syndrome of $(\mathbf{x}, \mathbf{y}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, denoted by $\mathbf{s}$, is computed as $\mathbf{s} = \mathbf{x} \cdot \mathbf{h}_1 + \mathbf{y} \cdot \mathbf{h}_2$. In this case, $\mathbf{h}_1$ and $\mathbf{h}_2$ are circulant matrices, while, $\mathbf{x}$ and $\mathbf{y}$ are vectors of Hamming weight w. Moreover, in HQC, systematic QC codes are used, hence $\mathbf{h}_1 = \mathbf{I}_n$. Consequently, the parity-check matrix that represents uniquely the public code used in HQC takes the form $\mathbf{H} = (\mathbf{I}_n, \mathbf{h})$ where $\mathbf{h}_2 = \mathbf{h}$.

Now, let's consider the extraction of a random binary symbol $b \in \{0, 1\}$. Then, if $b = 0$, the syndrome is computed as $\mathbf{s} = \mathbf{x} \cdot \mathbf{h}_1 + \mathbf{y} \cdot \mathbf{h}_2 = \mathbf{x} + \mathbf{y} \cdot \mathbf{h}$, whereas, if $b = 1$, let's assume $\mathbf{s}$ is randomly selected. At this point, it is easy to see that the parity of $\mathbf{s}$, i.e., whether $\mathbf{s}$ has an even or odd number of 1s, is closely related to w and the Hamming weight of $\mathbf{h}$, denoted as w_h. For example, let's assume that w_h is odd. From Theorem 4.2 and Theorem 4.3, it follows that:

- If w is odd, then $\mathbf{y} \cdot \mathbf{h}$ has an odd weight. Therefore, by adding it to $\mathbf{x}$ (also with an odd weight) to compute the syndrome $\mathbf{s}$, we obtain an even weight, thus $\mathbf{s}$ has an even weight.

- If w is even, then $\mathbf{y} \cdot \mathbf{h}$ has an even weight. Therefore, by adding it to $\mathbf{x}$ (also with an even weight) to compute the syndrome $\mathbf{s}$, we obtain an even weight, thus $\mathbf{s}$ has an even weight.

Thus, when $\mathbf{s}$ is randomly selected, it exhibits an approximately equal likelihood of having even or odd weight. However, when $\mathbf{s}$ represents an "authentic" syndrome (generated from the s-QCSD distribution), this balance is disrupted. In this scenario, an attacker who assesses the syndrome's parity can considerably discern whether it is a "legitimate" syndrome or not, effectively solving Problem 5 with non-negligible advantage.

To circumvent these trivial distinguishers, it is necessary to introduce an additional requirement regarding the syndrome's parity. In particular, for

$b \in \{0,1\}$, we define the finite set $\mathbb{F}_{2,b}^n = \{\mathbf{h} \in \mathbb{F}_2^n \text{ s.t. } \mathbf{h}(1) = b \mod 2\}$, i.e. binary vectors of length n and parity b. Indeed, $\mathbf{h}$ can be seen as a polynomial; therefore, $\mathbf{h}(1)$ represents the sum of a number of 1s equal to the weight of $\mathbf{h}$. Hence, if $\mathbf{h}$ has an even number of 1s, it follows that $b = 0$; otherwise, $b = 1$. To summarize:

- $b = 0 \rightarrow \mathbb{F}_{2,0}^n$ is the set of binary n-tuples with an even number of 1s.
- $b = 1 \rightarrow \mathbb{F}_{2,1}^n$ is the set of binary n-tuples with an odd number of 1s.

Similarly for matrices, we define the finite sets:

$$\mathbb{F}_{2,b}^{n\times 2n} = \{\mathbf{H} = (\mathbf{I}_n, \text{ rot}(\mathbf{h})) \in \mathbb{F}_2^{n\times 2n} \text{ s.t. } \mathbf{h} \in \mathbb{F}_{2,b}^n\}$$

$$\mathbb{F}_{2,b_1,b_2}^{2n\times 3n} = \left\{ \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \text{rot}(\mathbf{h}_1) \\ \mathbf{0} & \mathbf{I}_n & \text{rot}(\mathbf{h}_2) \end{pmatrix} \in \mathbb{F}_2^{2n\times 3n} \text{ s.t. } \mathbf{h}_1 \in \mathbb{F}_{2,b_1}^n \text{ and } \mathbf{h}_2 \in \mathbb{F}_{2,b_2}^n \right\}$$

At this point, what is done next is to rephrase the previously described problems by imposing a constraint on the parity of the syndrome. Specifically, it is required that the legitimate syndrome has the same parity as the random syndrome, thus preventing the presence of distinguishers that rely on evaluating the syndrome's parity.

Definition 4.5. *(s-QCSD Distribution with parity) For positive integers n, w, b_1, $\cdots$, b_{s-1} and s, the s-QCSD$(n, w, b_1, \cdots, b_{s-1})$ Distribution with parity chooses uniformly at random a parity-check matrix $\mathbf{H} \in \mathbb{F}_{2,b_1,\cdots,b_{s-1}}^{(sn-n)\times sn}$ of a QC code $\mathcal{C}$ of index s and rate $1/s$ together with a vector $\mathbf{x} = (\mathbf{x}_0, \cdots \mathbf{x}_{s-1}) \xleftarrow{\$} \mathbb{F}_2^{sn}$ such that $wt(x_i) = w$ for every $i = 0, \cdots s-1$ and returns in output $(\mathbf{H}, \mathbf{y}^T)$ with $\mathbf{y}^T = \mathbf{H} \cdot \mathbf{x}^T$.*

In particular, specializing to the case $s = 2$, we obtain:

Definition 4.6. *(2-QCSD Distribution with parity) For positive integers n, w, and b, the 2-QCSD(n, w, b) Distribution with parity chooses uniformly at*

random a parity-check matrix $\mathbf{H} \in \mathbb{F}_{2,b}^{n \times 2n}$ *of a QC code* $\mathcal{C}$ *of index 2 and rate* $1/2$ *together with a vector* $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1) \xleftarrow{\$} \mathbb{F}_2^{2n}$ *such that* $wt(x_0) = wt(x_1) = w$ *and outputs* $(\mathbf{H}, \mathbf{y}^T)$ *with* $\mathbf{y}^T = \mathbf{H} \cdot \mathbf{x}^T$.

At this point, it is possible to define the decisional SDP problem by assuming the use of quasi-cyclic codes and imposing a specific constraint on the syndrome to prevent the presence of distinguishers. By consolidating all the results we have seen so far, we obtain:

Problem 6. ***Decisional*** 2***-QCSD Problem with parity***
Let $\mathbf{h} \in \mathbb{F}_{2,b}^n$, $\mathbf{H} = (\mathbf{I}_n \ rot(\mathbf{h}))$ *and* $b' = w + b \times w \mod 2$. *For* $\mathbf{y} \in \mathbb{F}_{2,b'}^n$, *the Decisional 2-QCSD Problem with parity, denoted as 2-DQCSD*(n, w, b), *asks to decide with non-negligible advantage whether or not* $(\mathbf{H}, \mathbf{y})$ *came from the 2-QCSD*(n, w, b) *distribution with parity or the uniform distribution over* $\mathbb{F}_{2,b}^{n \times 2n} \times \mathbb{F}_{2,b'}^n$.

Specializing to the case $s = 3$, we obtain the following distribution and associated problem.

Definition 4.7. *(3-QCSD Distribution with parity) For positive integers* n, w, b_1 *and* b_2, *the 3-QCSD*(n, w, b_1, b_2) *Distribution with parity chooses uniformly at random a parity-check matrix* $\mathbf{H} \in \mathbb{F}_{2,b_1,b_2}^{2n \times 3n}$ *of a QC code* $\mathcal{C}$ *of index 3 and rate* $1/3$ *together with a vector* $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2) \xleftarrow{\$} \mathbb{F}_2^{3n}$ *such that* $wt(x_0) = wt(x_1) = wt(x_2) = w$ *and outputs* $(\mathbf{H}, \mathbf{y}^T)$ *with* $\mathbf{y}^T = \mathbf{H} \cdot \mathbf{x}^T$.

Problem 7. ***Decisional*** 3***-QCSD Problem with parity***
Let $\mathbf{h}_1 \in \mathbb{F}_{2,b_1}^n$, $\mathbf{h}_2 \in \mathbb{F}_{2,b_2}^n$, $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & rot(\mathbf{h}_1) \\ \mathbf{0} & \mathbf{I}_n & rot(\mathbf{h}_2) \end{pmatrix}$, $b'_1 = w + b_1 \times w \mod 2$ *and* $b'_2 = w + b_2 \times w \mod 2$. *For* $(\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{F}_{2,b'_1}^n \times \mathbb{F}_{2,b'_2}^n$, *the Decisional 3-QCSD Problem with parity, denoted as 3-DQCSD*(n, w, b_1, b_2), *asks to decide with non-negligible advantage whether or not* $(\mathbf{H}, (\mathbf{y}_1, \mathbf{y}_2))$ *came from the 3-QCSD*(n, w, b_1, b_2) *distribution with parity or the uniform distribution over* $\mathbb{F}_{2,b_1,b_2}^{2n \times 3n} \times (\mathbb{F}_{2,b'_1}^n \times \mathbb{F}_{2,b'_2}^n)$.

The sole distinction between Problem 6 and Problem 7 in comparison to Problem 5 is the imposition of a specific constraint on the syndrome's parity when it is randomly selected. This guarantees that whether the syndrome is legitimate (generated by the s-QCSD Distribution with parity) or random, it maintains the same parity, thus circumventing the previously described distinguishers.

The security of HQC under the IND-CPA (Indistinguishability under Chosen Plaintext Attack) relies significantly on the complexity of the 2 and 3-DQCSD problems (Problems 6 and 7). However, to enhance its resilience against structural attacks, it is essential to work with a code of primitive prime length. Nevertheless, for the chosen parameters and codes used in the HQC implementations, the size of the message encoding, represented as $\mathbf{m}$, is typically a product of two integers, i.e., $n_1 \cdot n_2$, which is not typically prime. To address this, we choose n as the ambient length, which is the first primitive prime number greater than $(n_1 \cdot n_2)$. We then truncate the last ℓ bits, where ℓ is equal to $n-(n_1 \cdot n_2)$, as needed. This modification results in a slightly altered version of the DQCSD problem. We contend that this modified problem is at least as challenging as the original one. To begin, we will define this truncated version in its primal form.

Problem 8. ***Decoding with ℓ erasures***

Let $\mathcal{C}[n,k]$ be a Quasi-Cyclic code generated by the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$. Consider a codeword $\mathbf{c} = \mathbf{mG} + \mathbf{e} \in \mathbb{F}_2^n$ in $\mathcal{C}$, where $\mathbf{m} \in \mathbb{F}_2^k$ is the information sequence and $\mathbf{e} \xleftarrow{\$} \mathbb{F}_{2,w}^n$. Here, $\mathbb{F}_{2,w}^n$ represents the set of all n-tuples over $\mathbb{F}_2$ with a Hamming weight of w. Now, consider the matrix $\mathbf{G'} \in \mathbb{F}_2^{k \times n'}$, obtained by removing the last $\ell = n - n' \geqslant 1$ columns from $\mathbf{G}$ and the vector $\mathbf{e'} \in \mathbb{F}_2^{n'}$, obtained by removing the last $\ell = n - n' \geqslant 1$ columns from $\mathbf{e}$. The decoding with ℓ erasures problem aims to recover $\mathbf{m} \in \mathbb{F}_2^k$ from $\mathbf{c'} = \mathbf{mG'} + \mathbf{e'} \in \mathbb{F}_2^{n'}$ and $\mathbf{G'} \in \mathbb{F}_2^{k \times n'}$.

This problem seeks to recover the encoded message, given less informa-

tion. It then becomes evident that Decoding with erasures is inherently more challenging than decoding with complete knowledge of the encoding. Suppose we assume that $\mathcal{A}$ can successfully solve the decoding problem with ℓ erasures, and let $(\mathbf{c}, \mathbf{G})$ represent an instance of the decoding problem without any erasure. The approach starts by eliminating the last ℓ columns from both $\mathbf{c}$ and $\mathbf{G}$, and then employing $\mathcal{A}$ to retrieve $\mathbf{m} \in \mathbb{F}_2^k$. Given that the dimension remains unchanged in both problems, $\mathbf{m}$ is also a solution to the decoding problem without any erasures, thus confirming its level of difficulty. Because the decoding problem and the syndrome decoding problem are equivalent (as proofed in Theorem 4.1), the previously outlined argument remains applicable. Consequently, the corresponding 2 and 3-DQCSD problems with $\ell = n - n_1 n_2$ erasures, tailored to counter structural attacks, are at least as hard as those defined in Problem 6 and Problem 7.

Assumption 1. *While there is no comprehensive complexity result for quasi-cyclic codes, the decoding of these codes is widely recognized as a challenging task within the community. General attacks that leverage the cyclic structure of the code have been proposed [17], but their impact on the problem's complexity is relatively minor (sub-linear with respect to the code length). In practical terms, it is observed that the most effective attacks closely resemble those used for non-circulant codes, with only a slight variation.*

To sum up, it's crucial to emphasize that HQC provides IND-CPA security, assuming the hardness of 2-QCSD with parity and 3-QCSD with parity and erasures. This outcome is proven in [18].

4.3 Complexity classes

In order to evaluate the potential for building a cryptographic system on a specific problem, like SDP, it is crucial to determine its inherent complexity. To accomplish this, we offer a concise introduction to the fundamental principles

of complexity theory [19]. It's important to mention that the subsequent discussion will focus on presenting these essential concepts in an intuitive manner, rather than delving into rigorous details.

Definition 4.8. *P denotes the set of decisional problems that can be efficiently solved in polynomial time with respect to the size of the input.*

A problem is categorized as belonging to P if there exists at least one algorithm capable of solving it within polynomial time. For instance, tasks like determining whether a number n is even or odd or establishing, given an array of n integers, whether an integer k is a part of it or not fall within the complexity class P. In essence, problems within P are considered easy to solve.

Definition 4.9. *NP denotes the class of decisional problems for which solutions can be efficiently verified in polynomial time with respect to the size of the input.*

Solving many of these problems requires exponential time. Examples of problems in NP are:

1. SAT (Satisfaiability): it is the problem of determining whether a Boolean formula is satisfiable or unsatisfiable. In other words, it asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. In this case, once a solution is given, it is extremely easy to verify its validity. On the other hand, for non-trivial instances of SAT, determining a solution is extremely complex.

2. VertexCover: in the context of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a vertex cover is a subset of vertices $\mathcal{V}^* \subseteq \mathcal{V}$ where, for every edge $e = uv \in \mathcal{E}$, the intersection of $\mathcal{V}^*$ with the set $\{u, v\}$ is not empty. Here, $\mathcal{E}$ represents the set of edges, and $\mathcal{V}$ represents the set of vertices within the graph $\mathcal{G}$. In simpler

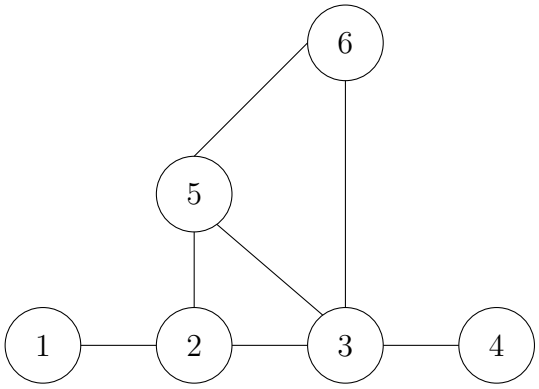

Figure 4.3: Example of a VertexCover instance

terms, a vertex cover in a graph is a set $\mathcal{V}^*$ of nodes where every edge either originates from or terminates at one of the nodes in the set. The problem involves a graph $\mathcal{G}$ and an integer k as input, and it is expressed as follows: "Does there exist a vertex cover in $\mathcal{G}$ with a cardinality of k?" To illustrate this concept, consider a straightforward example as shown in Figure 4.3. If we set $k = 3$, it becomes clear that there is indeed a set of three nodes in which every edge in the graph connects to or emanates from this specific set. For instance, a valid vertex cover could include nodes 2, 3, and 5.

3. SetCover: given a set U of elements and a set $\{S_1, \cdots, S_n\}$ of subsets of U, the set cover problem aims to determine if there exist k subsets that cover the entire set U. For example, if we consider $U = \{1, 2, 3, 4, 5, 6, 7\}$ and subsets $S_1 = \{1\}$, $S_2 = \{1, 2, 3\}$, $S_3 = \{3, 4, 5, 6\}$, $S_4 = \{5\}$, $S_5 = \{2, 4, 7\}$ and $S_6 = \{6, 7\}$, assuming $k = 3$, one possible solution can be represented by S_2, S_3 and S_6. In addition, another possible solution for this SetCover instance is S_2, S_3 and S_5.

Generally, for SAT, VertexCover, and SetCover, once a solution is provided, confirming its correctness is straightforward. In the instances of VertexCover and SetCover we have examined so far, their simplicity allows for easy solution determination. Nevertheless, this generalization doesn't hold, particularly for

"large" instances, where finding solutions becomes considerably challenging.

Clearly, a problem that is easy to solve is also easy to verify because, in fact, solving it is sufficient for verifying its solutions. This justifies why the complexity class P is contained within the complexity class NP. Problems in NP might potentially belong to P, but efficient algorithms for solving them have not yet been discovered. Over the years, the study of the relationship between P and NP has focused on identifying the "most difficult" problems within the NP class. At this point, it becomes crucial to determine when a problem $\mathcal{X}$ is easier to solve than another problem $\mathcal{Y}$. In this regard, it is initially necessary to define the concept of polynomial-time reductions.

Definition 4.10. *(Polynomial-time reduction) Consider two decisional problems, $\mathcal{X}$ and $\mathcal{Y}$. A polynomial-time reduction from $\mathcal{X}$ to $\mathcal{Y}$ encompasses the following steps:*

1. *take an instance I of $\mathcal{X}$;*
2. *transform I into an instance I' of $\mathcal{Y}$ in polynomial-time;*
3. *assume that $\mathcal{Y}$ can be solved in the instance I' and its solution is s';*
4. *transform s' into a solution s for $\mathcal{X}$ in the instance I in polynomial time.*

A fundamental prerequisite is that this transformation should not mix "YES" instances with "NO" instances.

In this scenario (depicted in Figure 4.4), if we were capable of solving $\mathcal{Y}$, we would also be able to solve $\mathcal{X}$ by simply transforming an instance and a solution. Consequently, it can be inferred that $\mathcal{Y}$ presents at least a higher level of complexity than problem $\mathcal{X}$. The concept of polynomial-time reduction enables us to delineate the class of problems that are of particular interest from a post-quantum security perspective.

Definition 4.11. *NP-hard denotes the class of decisional problems for which there exists a polynomial-time reduction from every problem in NP.*

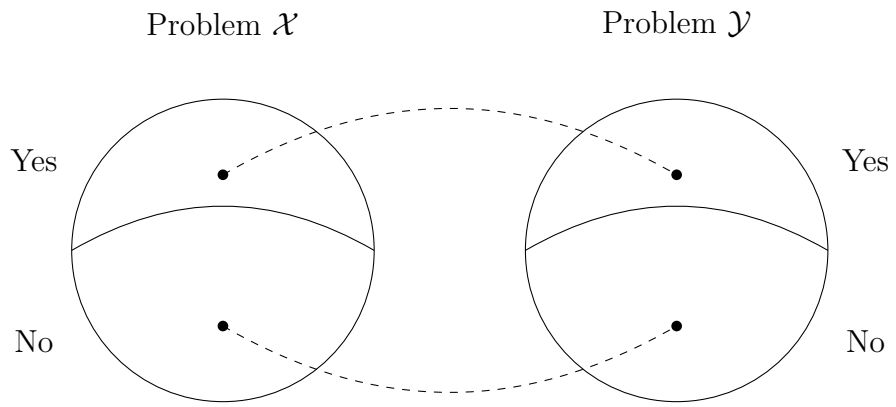

Figure 4.4: Reduction from $\mathcal{X}$ to $\mathcal{Y}$

According to the concept of polynomial-time reduction (Definition 4.10), NP-hard problems are recognized as being at least as challenging as any problem in NP. Therefore, NP-hard problems are considered among the most difficult problems to solve and, for this reason, they are of particular interest in cryptography. To establish that a problem, denoted as $\mathcal{Z}$, is NP-hard, it may seem necessary, according to the definition, to demonstrate that every problem in NP is reducible in polynomial-time to $\mathcal{Z}$. However, proving that $\mathcal{Z}$ is NP-hard doesn't require such an exhaustive approach. Instead, this proof can be achieved by demonstrating that $\mathcal{Z}$ can be reduced in polynomial-time to a problem already known to belong to the NP-hard class. For instance, the integer factorization problem is in NP because given a proposed set of prime factors for the composite integer, it is straightforward to verify whether their product equals the original integer in polynomial time. However, despite being a challenging problem, it is not known to be NP-hard, which means there is no known polynomial-time reduction from an NP-hard problem to the integer factorization problem. In this section, we have primarily focused on decisional problems. It's worth noting that an attacker attempting to breach a cryptographic scheme must solve the problem in its computational version. Intuitively, it's clear that a problem in its computational version is at least as difficult as its counterpart in the decisional version. Therefore, a decisional

problem within NP-hard has its corresponding computational problem also within NP-hard. This basic consideration is highly significant when considering these problems as foundational components of public key cryptographic schemes. Indeed, NP-hard problems are believed to be quantum-resistant, making them suitable as underlying problems for cryptographic schemes. A problem that resides in NP-hard is obviously extremely challenging to solve, nevertheless, in general, a problem of this nature is also complex to verify. In other words, typically, when given a solution for an NP-hard problem, establishing the validity of the solution can be intricate. Therefore, to summarize, an NP-hard problem does not necessarily fall within the previously defined complexity class NP. The Halting Problem is a classic example of a problem that is NP-Hard but not in NP. It involves determining whether a given computer program will halt or run indefinitely for a given input. It's undecidable [20], indicating that there's no universal algorithm to solve it for all program-input combinations. It can't be in NP because there's no known polynomial-time verification method, but it is NP-hard because it can be reduced to other well-known NP-hard problems, such as SAT. Clearly, when considering a problem as a mathematical foundation for a cryptographic scheme, there might be a need for problems that possess a dual characteristic: they are exceptionally hard to solve on one hand, but on the other hand, once a solution is provided, it is straightforward to verify its correctness. To address this need, an additional class of problems is defined, known as NP-complete.

Definition 4.12. *NP-complete is the class that results from the intersection of NP-hard and NP classes.*

Problems in the NP-complete class are extremely difficult because they are NP-hard, and in addition, given a solution, it is easy to verify its validity as they are in NP. This dual characteristic makes them ideal for cryptographic purposes. Cook demonstrated the NP-completeness of SAT in [21]. Garey and Johnson proofed the NP-completeness of VertexCover in [22] and Karp showed the

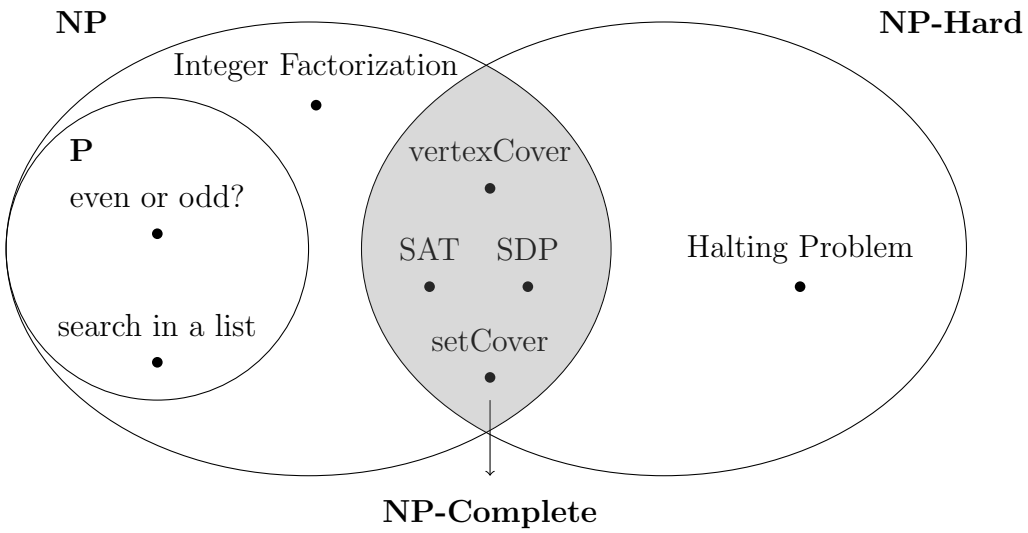

Figure 4.5: Complexity Classes and associated problems.

NP-completeness of SetCover in [23]. Separately, Berlekamp, McEliece, and Van Tilborg provided evidence for the NP-completeness of SDP for binary linear codes using the Hamming metric (Definition 2.6), as documented in [24]. Additionally, Barg extended this demonstration to finite fields of various sizes, as outlined in [25]. These findings firmly establish SDP as a compelling candidate for the underlying problem in a post-quantum cryptographic scheme. A graphical representation of various complexity classes with some of the most common problems is shown in Figure 4.5.

As previously stated, the security of HQC relies on Problem 6 and Problem 7 combined with Problem 8. However, even though the SDP problem has been proven to be NP-complete for linear codes, there is no analogous proof for quasi-cyclic codes. Nevertheless, what is stated in Assumption 1 holds true.

Chapter 5

Description of HQC

HQC (Hamming Quasi Cyclic) is based on an encryption scheme comprising four polynomial-time algorithms (Setup, KeyGen, Encrypt, Decrypt):

- Setup(1^λ): output the global parameters denoted as **param** of the scheme. In this case λ is the security parameter;

- KeyGen(**param**): produces a pair of keys, specifically a public encryption key denoted as pk and a private decryption key denoted as sk. The key pk is public, therefore known to everyone. The key sk is secret and known only to the legitimate user, i.e. the user who legitimately receives the message encrypted with the corresponding public key.

- Encrypt($\mathsf{pk}, \mathbf{m}, \theta$): yields a ciphertext denoted as $\mathbf{c}$ for the message $\mathbf{m}$ using the encryption key pk and the randomness θ. For the sake of clarity, we also employ the notation Encrypt($\mathsf{pk}, \mathbf{m}$);

- Decrypt($\mathsf{sk}, \mathbf{c}$): retrieves the plaintext $\mathbf{m}$ corresponding to the ciphertext $\mathbf{c}$ or outputs $\perp$ in case of decoding failure.

An encryption scheme must meet the stringent criteria of both Correctness and Indistinguishability under a Chosen Plaintext Attack (IND-CPA) to be deemed secure and reliable.

Definition 5.1. *(Correctness) An encryption scheme satisfies the Correctness property when the probability that Decrypt*$(\mathsf{sk}, \mathit{Encrypt}(\mathsf{pk}, \mathbf{m}, \theta))$ *equals* $\mathbf{m}$ *is* $1 + \mathit{negl}(\lambda)$ *with* λ *representing the security parameter.*

Correctness ensures that, for every chosen security parameter λ, every set of parameters generated during the Setup process, and for each pair of keys produced by the KeyGen algorithm, the encryption and decryption operations must consistently deliver the desired outcome. In other words, for any given message $\mathbf{m}$ encrypted with the public key pk and subsequently decrypted using the private key sk, the result should be equal to the original message $\mathbf{m}$.

A cryptosystem is deemed secure in terms of indistinguishability if no adversary, when given an encryption of a message randomly selected from a two-element message space determined by the adversary, can identify the message choice with a significantly higher probability than random guessing. If any adversary manages to distinguish the chosen ciphertext with a probability significantly exceeding 1/2, it is considered to have an "advantage" in distinguishing the ciphertext, and the scheme is not regarded as secure in terms of indistinguishability. This definition implies that the adversary should extract no information from observing a ciphertext. As a result, the adversary should not fare any better than random guessing when attempting to distinguish the message from the ciphertext. The concept of security based on indistinguishability spans across various definitions, with the variations rooted in the assumptions about the attacker's capabilities. This multifaceted security paradigm is often illustrated as a game, serving as a fundamental tool to evaluate the robustness of cryptosystems. In essence, the cryptosystem is deemed secure when no adversary can attain a substantially greater probability of winning the game than an adversary who relies on random guessing.

IND-CPA security ensures that even when an adversary is aware that one of two self-selected messages has been encrypted, he cannot efficiently discern which one. This security is defined and assessed through the following game:

IND-CPA game

1. The challenger creates a key pair using the KeyGen algorithm, sharing the public key pk with the adversary while keeping the secret key sk.

2. The adversary is allowed to carry out a limited number of encryptions, constrained by a polynomial limit.

3. The adversary provides two different chosen plaintexts, M_0 and M_1, to the challenger.

4. The challenger randomly selects a bit $b = \{0, 1\}$ and encrypts the chosen plaintext M_b, resulting in the challenge ciphertext $\mathbf{c} = \text{Encrypt}(\mathsf{pk}, \mathsf{M_b})$, which is then sent to the adversary;

5. The adversary is free to engage in further computations or encryptions;

6. Ultimately, the adversary makes a guess regarding the value of b.

Definition 5.2. *(IND-CPA security) A cryptosystem is considered IND-CPA secure when any probabilistic polynomial-time adversary has only a negligible advantage over random guessing. This negligible advantage is defined as the adversary winning the IND-CPA game with a probability of* $1/2 + negl(\lambda)$.

In addition to IND-CPA, it is necessary to define IND-CCA security, which encompasses IND-CCA1 and IND-CCA2. These refer to Indistinguishability under non-adaptive and adaptive Chosen Ciphertext Attack, respectively. In IND-CCA1 and IND-CCA2, the adversary has access to decryption alongside encryption. The non-adaptive version allows decryption queries until the challenge ciphertext is received. In the adaptive version, decryption queries continue but not for the challenge ciphertext to prevent trivial attacks. Games are used to assess and formalize security in these contexts, much like IND-CPA.

In this regard, the IND-CCA game is formalized below:

IND-CCA game

1. The challenger creates a key pair using the KeyGen algorithm, sharing the public key pk with the adversary while keeping the secret key sk;
2. The adversary may perform any number of calls to the encryptions and decryption oracle;
3. The adversary provides two different chosen plaintexts, M_0 and M_1, to the challenger;
4. The challenger randomly selects a bit $b = \{0, 1\}$ and encrypts the chosen plaintext M_b, resulting in the challenge ciphertext $\mathbf{c} = \text{Encrypt}(\mathsf{pk}, M_b)$, which is then sent to the adversary;
5. The adversary is free to perform any number of additional encryptions or decryptions:
   - In the non-adaptive case (IND-CCA1), the adversary may not make further calls to the decryption oracle.
   - In the adaptive case (IND-CCA2), the adversary may make further calls to the decryption oracle, but may not submit the challenge ciphertext $\mathbf{c}$.
6. Ultimately, the adversary makes a guess regarding the value of b.

Definition 5.3. *(IND-CCA1 and IND-CCA2 security) A cryptosystem is considered IND-CCA1 (IND-CCA2) secure when any probabilistic polynomial-time adversary has only a negligible advantage over random guessing. This negligible advantage is defined as the adversary winning the IND-CCA1 (IND-CCA2) game with a probability of* $1/2 + negl(\lambda)$.

As discussed in Section 4.2, HQC has been proved IND-CPA secure. However, the standard (highest) security requirement for a public key cryptosystem is IND-CCA2. In this regard, HQC's authors used the Fujisaki-Okamoto Transformation [26] to move from a secure IND-CPA Public Key Encryption to a secure IND-CCA2 Key Encapsulation Mechanism.

5.1 HQC as a Public Key Encryption

We recall the key generation and encryption algorithms for HQC. Specifically, $\mathbf{G}$ denotes the generator matrix of a $[n,k]$ code $\mathcal{C}$ with decoding algorithm $\mathfrak{D}$.

HQC.PKE

Setup(1^λ): generates the global parameters $\mathbf{param} = (n, k, \delta, w, w_r, w_e)$

Key Generation through the KeyGen($\mathbf{param}$) algorithm:

1. sample $\mathbf{h} \xleftarrow{\$} \mathcal{R}$;
2. sample $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{R}(w)$;
3. set $\mathsf{sk} := (\mathbf{x}, \mathbf{y})$ and $\mathsf{pk} := \{\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{hy}\}$.

Encryption through the Encrypt($\mathsf{pk}, \mathbf{m}$) algorithm:

1. sample $\mathbf{r}^{(1)}, \mathbf{r}^{(2)} \xleftarrow{\$} \mathcal{R}(w_r)$, $\mathbf{e} \xleftarrow{\$} \mathcal{R}(w_e)$;
2. set $\mathbf{u} = \mathbf{r}^{(1)} + \mathbf{hr}^{(2)}$;
3. set $\mathbf{v} = \mathbf{mG} + \mathbf{e} + \mathbf{sr}^{(2)}$;
4. the ciphertext is $\{\mathbf{u}, \mathbf{v}\}$.

Decryption through the Decrypt($\mathsf{sk}, (\mathbf{u}, \mathbf{v})$) algorithm:

1. set $\mathbf{c} = \mathbf{v} + \mathbf{y} \cdot \mathbf{u}$;
2. run $\mathfrak{D}(\mathbf{c})$.

Let's consider $\mathbf{c} = \mathbf{v} + \mathbf{y} \cdot \mathbf{u}$. Substituting the definitions of $\mathbf{u}$ and $\mathbf{v}$ into this expression, we arrive at:

$$\mathbf{c} = \mathbf{v} + \mathbf{y} \cdot \mathbf{u} = \mathbf{mG} + \mathbf{e} + \mathbf{sr}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{y} \cdot \mathbf{hr}^{(2)} \tag{5.1}$$

Given that $\mathbf{s} = \mathbf{x} + \mathbf{hy}$, we can deduce:

$$\mathbf{hy} = \mathbf{s} - \mathbf{x} = \mathbf{s} + \mathbf{x} \tag{5.2}$$

By substituting equation (5.2) into (5.1), we get:

$$\mathbf{c} = \mathbf{mG} + \mathbf{e} + \mathbf{sr}^{(2)} + \mathbf{yr}^{(1)} + \mathbf{r}^{(2)}(\mathbf{s} + \mathbf{x}) \tag{5.3}$$

Simplifying further:

$$\mathbf{c} = \mathbf{mG} + \mathbf{e} + \mathbf{yr}^{(1)} + \mathbf{xr}^{(2)} \tag{5.4}$$

Let's assume:

$$\mathbf{z} = \mathbf{e} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{x} \cdot \mathbf{r}^{(2)} \tag{5.5}$$

which leads us to:

$$\mathbf{c} = \mathbf{mG} + \mathbf{z} \tag{5.6}$$

The correctness of the encryption scheme is directly dependent on the decoding capability of the code $\mathcal{C}$. In particular, assuming that $\mathfrak{D}$ accurately deciphers $\mathbf{c} = \mathbf{mG} + \mathbf{z}$, the following relationship holds:

$$\text{Decrypt}(\mathsf{sk}, \text{Encrypt}(\mathsf{pk}, \mathbf{m})) = \mathbf{m}.$$

Specifically, decoding is successful when the Hamming weight of the resulting error pattern $\mathbf{z}$ is less than the corrective power of the code, in symbols:

$$\text{wt}(\mathbf{z}) = \text{wt}(\mathbf{e} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{x} \cdot \mathbf{r}^{(2)}) \leqslant \delta$$

Furthermore, it is important to emphasize that the term $\mathbf{v}$ represents a codeword ($\mathbf{mG}$) that has been affected by noise contamination. Specifically, the sparse nature of the term $\mathbf{e}$ (with $w_e \ll n$) implies that it contains relatively few errors. In the absence of the term $\mathbf{sr}^{(2)}$ within $\mathbf{v}$, retrieving $\mathbf{m}$ from the codeword would be a relatively straightforward task, potentially compromising the confidentiality of HQC as a Public Key Encryption (PKE) scheme. For this reason, the polynomial $\mathbf{s}$ is crucial and takes on the meaning of syndrome. Given that $\mathbf{s}$ plays the role of a syndrome, to maintain its indistinguishability from a randomly generated syndrome, it must exhibit an average weight of around 50%. This requirement results in $\mathbf{sr}^{(2)}$ introducing errors into roughly half of the symbols. The introduction of $\mathbf{sr}^{(2)}$ in the expression of $\mathbf{v}$ carries significant implications. First of all, decoding complexity underscores the key factor that this process is only feasible when the secret key sk (trapdoor) is known. Specifically, when sk is known, it becomes possible to compute $\mathbf{z}$ as detailed in (5.5). In this context, decoding becomes straightforward due to the sparsity of $\mathbf{z}$. Notably, based on Theorems 4.2 and 4.3, we can deduce that the maximum Hamming weight of $\mathbf{z}$ is $w \cdot w_r + w \cdot w_r + w_e$ (neglecting hypothetical cancellations). Since $\mathbf{x}, \mathbf{y}, \mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ are by hypothesis sparse, it follows that $w, w_r, w_e \ll n$, therefore, $\mathbf{z}$ is sparse itself.

5.2 HQC as a Key Encapsulation Mechanism

Suppose we have an instance of HQC.PKE denoted as $\mathcal{E}$. Additionally, let $\mathcal{G}$, $\mathcal{H}$, and $\mathcal{K}$ represent hash functions, such as SHA-512 [27]. Briefly, a cryptographic hash function is a surjective function that takes a message of any length and returns a fixed-length message digest. To be considered a reliable hash function, it must meet several key criteria. Firstly, it should be computationally fast to calculate. Secondly, it must exhibit one-way resistance (pre-image resistance), meaning it should be practically impossible to reverse

the process and retrieve the original input from the digest. Additionally, it should be weakly resistant to collisions (second pre-image resistance), which means that given a fixed input, finding another input that produces the same digest should be extremely challenging. Lastly, the hash function should also be strongly collision-resistant, implying that finding two different inputs that yield the same digest should be computationally infeasible.

The KEM-DEM version of HQC is defined as follows:

HQC.KEM

Setup(1^λ): as for HQC.PKE, with the only difference being that k represents the length of the symmetric key being exchanged;

Key Generation as for HQC.PKE;

Encapsulate with input the public key pk:

1. generate $\mathbf{m} \in \mathbb{F}_2^k$ as a seed to derive the shared key K;
2. set the randomness $\theta = \mathcal{G}(\mathbf{m})$;
3. generate the ciphertext $\mathbf{c} = \mathcal{E}.\text{Encrypt}(\mathsf{pk}, \mathbf{m}, \theta)$
4. the shared key is $K = \mathcal{K}(\mathbf{m}, \mathbf{c})$;
5. set $\mathbf{d} = \mathcal{H}(\mathbf{m})$ and send $(\mathbf{c}, \mathbf{d})$.

Decapsulate with input the secret key sk, $\mathbf{c}$ and $\mathbf{d}$:

1. decrypt $\mathbf{m}' = \mathcal{E}.\text{Decrypt}(\mathsf{sk}, \mathbf{c})$
2. compute $\theta' = \mathcal{G}(\mathbf{m}')$
3. re-encrypt $\mathbf{m}'$ to get $\mathbf{c}' = \mathcal{E}.\text{Encrypt}(\mathsf{pk}, \mathbf{m}', \theta')$
4. if $\mathbf{c} = \mathbf{c}'$ and $\mathbf{d} = \mathcal{H}(\mathbf{m}')$, then obtain the shared key $K = \mathcal{K}(\mathbf{m}, \mathbf{c})$; otherwise, abort.
5. once K is exchanged, it can be used to encrypt with a symmetric cryptosystem in a hybrid encryption context.

First, it is reasonable to question why we transmit $\mathbf{d} = \mathcal{H}(\mathbf{m})$ instead of directly employing $\theta = \mathcal{G}(\mathbf{m})$. If we were to employ $\theta = \mathcal{G}(\mathbf{m})$ directly, rather than $d = \mathcal{H}(\mathbf{m})$, it would expose the randomness to the public. The randomness θ plays a vital role during the encryption phase, as it dictates the selection of the polynomials $\mathbf{r}^{(1)}, \mathbf{r}^{(2)} \xleftarrow{\$} \mathcal{R}(w_r)$ and $\mathbf{e} \xleftarrow{\$} \mathcal{R}(w_e)$. Considering that, during encryption, $\mathbf{v} = \mathbf{mG} + \mathbf{sr}^{(2)} + \mathbf{e}$ becomes public, once $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}$, and $\mathbf{e}$ are known (since the randomness θ becomes public), given that $\mathbf{s}$ is known since it's part of the public key pk and since $\mathbf{G}$ is public, the only remaining unknown in $\mathbf{v}$ expression is $\mathbf{m}$, which can then be determined. Consequently, the attacker can recover the seed for the shared key, enabling the computation of $K = \mathcal{K}(\mathbf{m}, \mathbf{c})$ and compromising the scheme's confidentiality.

5.3 Settings and Parameters

We here recall the HQC versions ([28], [29], [30] and [31]) which have been submitted to NIST competition, together with the recommended parameters.

5.3.1 1st round submission

In the first round of the NIST submission, the public code $\mathcal{C}$ has been chosen as a tensor product code (defined in subsection 3.5.1), obtained from two binary codes $\mathcal{C}_1 \subseteq \mathbb{F}_2^{n_1}$ and $\mathcal{C}_2 \subseteq \mathbb{F}_2^{n_2}$. Namely, the authors of HQC chose $\mathcal{C}_1$ as a BCH(n_1, k, t_{BCH}) code (section 3.2) and $\mathcal{C}_2$ as a repetition code (section 3.1) of length n_2 and dimension $k_2 = 1$. In the context of the Hamming metric version of the cryptosystem, an initial message, denoted as $\mathbf{m}$, drawn from $\mathbb{F}_2^k$, undergoes a two-step encoding procedure. First, it is transformed into $\mathbf{m}_1 \in \mathbb{F}_2^{n_1}$ through a generator for $\mathcal{C}_1$. Subsequently, every symbol within this codeword undergoes encoding through the repetition code $\mathcal{C}_2$. In effect, each symbol of $\mathbf{m}_1$ is replicated a total of n_2 times, further enhancing the error-correcting capabilities of the tensor product code. Therefore, a codeword of

Code	n	k	t_{BCH}
BCH-1	1023	513	57
BCH-2	1023	483	60
BCH-S1	766	256	57
BCH-S2	796	256	60

Table 5.1: BCH codes in their original and shortened forms. The considered BCH codes are initially of length 1023, then shortened to support 256 bits dimension. It's important to note that the error-correcting capability of the code remains unaffected by the shortening operation.

the tensor product code $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$ has the following structure:

$$\big(\underbrace{c_0^{(0)}, c_1^{(0)}, \cdots, c_{n_2-1}^{(0)}}_{\text{Length } n_2}, \underbrace{c_0^{(1)}, c_1^{(1)}, \cdots, c_{n_2-1}^{(1)}}_{\text{Length } n_2}, \cdots, \underbrace{c_0^{(n_1-1)}, c_1^{(n_1-1)}, \cdots, c_{n_2-1}^{(n_1-1)}}_{\text{Length } n_2}\big)$$

We adopt the notation $n = n_1 \cdot n_2$ to denote the length of the tensor product code. In practice, the chosen length is the smallest primitive prime greater than n to enhance security against algebraic attacks (as described in section 4.2). Meanwhile, the dimension of the tensor product code is expressed as $k = k_1 k_2 = k_1$. Additionally, it's noteworthy that the minimum distance of the tensor product code is equal to the product of the minimum distances of its constituent codes.

Depending on the chosen parameters of the HQC scheme, shortened BCH codes are utilized (detailed in subsection 2.2.3). Specifically, the shortening process is applied to yield a BCH code with a dimension of 256. This operation is carried out by taking a BCH code, denoted as BCH-1$[1023,\ 513,\ t_{\text{BCH}} = 57]$, and transforming it into BCH-S1 by setting $\ell = 257$. Similarly, another BCH code, originally labeled as BCH-2 $[1023,\ 483,\ t_{\text{BCH}} = 60]$, is transformed into BCH-S2 with $\ell = 227$. Table 5.1 offers a comprehensive comparative analysis between BCH-1 and BCH-2 and their corresponding shortened variants, denoted as BCH-S1 and BCH-S2. The recommended parameters for this specific version of HQC are presented in Table 5.2.

SL	t_{BCH}	k	n_1	n_2	n	w	w_r	w_e	$P_{decoding\,failure}$
128	60	256	796	31	24,677	67	77	77	$< 2^{-128}$
192	57	256	766	57	43,669	101	117	117	$< 2^{-128}$
192	57	256	766	61	46,747	101	117	117	$< 2^{-192}$
256	57	256	766	83	63,587	133	153	153	$< 2^{-128}$
256	60	256	796	85	67,699	133	153	153	$< 2^{-192}$
256	60	256	796	89	70,853	133	153	153	$< 2^{-256}$

Table 5.2: HQC parameters for the Round 1 submission. Only the parameters to obtain DFR less than 2^{-128} are reported.

SL	t_{BCH}	k	n_1	n_2	n	w	w_r	w_e	$P_{decoding\,failure}$
128	57	256	766	31	23,869	67	77	77	2^{-128}
192	57	256	766	59	45,197	101	117	117	2^{-192}
256	60	256	796	87	69,259	133	153	153	2^{-256}

Table 5.3: HQC parameters for the Round 2 submission assuming that the public code can be expressed as $\mathcal{C} = \text{BCH}(n_1, k, t_{\text{BCH}}) \otimes \mathbf{1}_{n_2}$, where $\mathbf{1}_{n_2}$ denotes a repetition code of length n_2.

5.3.2 2nd round submission

In the second round of the NIST submission, HQC's authors introduced an additional decoding algorithm rooted in the concatenation of Reed-Muller and Reed-Solomon codes. Consequently, the second round encompasses two distinct approaches which provide two distinct sets of parameters targeting different level of Security Level (SL). The first approach doesn't warrant further exploration, as it mirrors the one employed in the first round of HQC. In this scenario, parameter sets are reported in Table 5.3. Meanwhile, the second approach, which incorporates Reed-Muller and Reed-Solomon codes, deserves further exploration as it is the new addition compared to the round 1 submission. This approach was introduced with the specific goal of reducing the key size. In this scenario, the public code $\mathcal{C}$ is assumed to be a concatenated code (described in section 3.5). It is formed by combining a Reed-Solomon code, discussed in subsection 3.3.1, as the outer code, and a first-order Reed-Muller code, detailed in subsection 3.4.1, which serves as the inner code. Figure 5.1 provides a visual representation of the typical structure of a concatenated code.

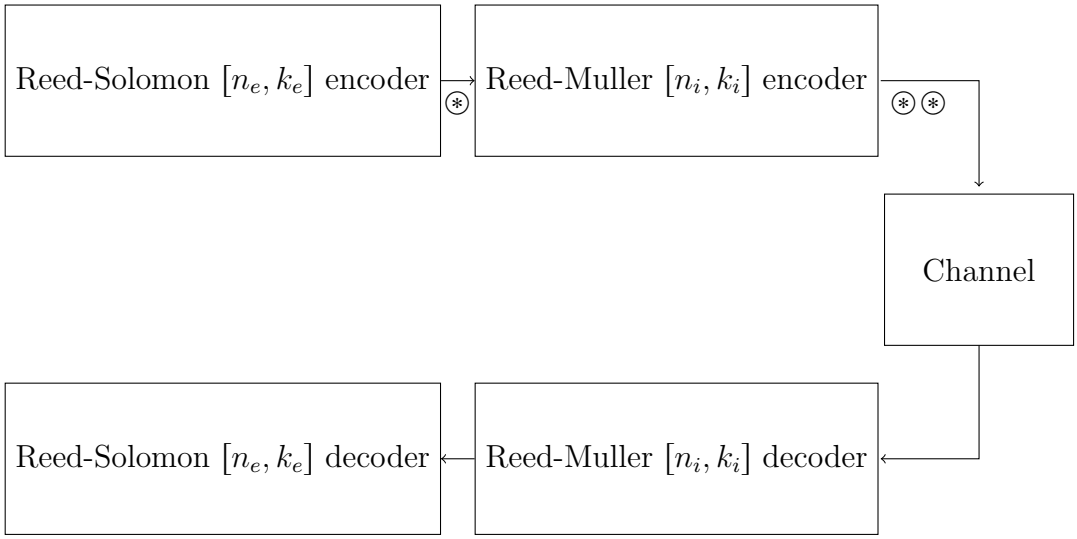

Figure 5.1: Reed-Solomon and Reed-Muller concatenated codes. $\circledast$ and $\circledast\circledast$ will serve to formalize the subjects at play in that specific section, specifically, in equations (5.8) and (5.9)

We start with an information sequence $\mathbf{m}$ consisting of k_e q-ary symbols:

$$\mathbf{m} = \left(m_0^{(0)} \cdots m_{k_i-1}^{(0)},\, m_0^{(1)} \cdots m_{k_i-1}^{(1)} \cdots m_0^{(k_e-1)} \cdots m_{k_i-1}^{(k_e-1)}\right) \in \mathbb{F}_q^{k_e} = \mathbb{F}_{2^{k_i}}^{k_e} \quad (5.7)$$

We have k_e q-ary symbols, where $q = 2^{k_i}$. As a result, each symbol can be represented as a k_i-tuple over $\mathbb{F}_2$, which means we have a total of $k_e k_i$ binary symbols. It's important to note that the Reed-Solomon encoder operates on q-ary symbols, as RS codes are not binary codes. Consequently, after the RS encoding process, we transition from having k_e q-ary symbols to n_e q-ary symbols, each of which can be represented using k_i binary symbols. Therefore:

$$\circledast : \mathbf{c}' = \left(c'^{(0)}_0 \cdots c'^{(0)}_{k_i-1},\, c'^{(1)}_0 \cdots c'^{(1)}_{k_i-1} \cdots c'^{(n_e-1)}_0 \cdots c'^{(n_e-1)}_{k_i-1}\right) \in \mathbb{F}_q^{n_e} = \mathbb{F}_{2^{k_i}}^{n_e} \quad (5.8)$$

In contrast, the Reed-Muller encoding operates on binary symbols. Consequently, we transition from blocks of k_i binary symbols (representing a q-ary symbol) to blocks of n_i binary symbols. Therefore, following the RM encoding, we have n_e blocks, each consisting of n_i binary symbols. Consequently,

Code	n	k	t_{RS}
RS-1	255	207	24
RS-2	255	211	22
RS-3	255	209	23
RS-S1	80	32	24
RS-S2	76	32	22
RS-S3	78	32	23

Table 5.4: Original and Shortened RS codes. The shortened codes RS-S1, RS-S2, and RS-S3 are obtained by assuming $\ell = 175$, $\ell = 179$, and $\ell = 177$ for RS-1, RS-2, and RS-3, respectively.

the total number of binary symbols is given by $n = n_e \cdot n_i$. Therefore:

$$\circledast\circledast : \mathbf{c}'' = \left(c''^{(0)}_{0} \cdots c''^{(0)}_{n_i-1},\, c''^{(1)}_{0} \cdots c''^{(1)}_{n_i-1} \cdots c''^{(n_e-1)}_{0} \cdots c''^{(n_e-1)}_{n_i-1}\right) \in \mathbb{F}_2^{n_e n_i} \quad (5.9)$$

The outer code selected for this configuration is a Reed-Solomon code over $\mathbb{F}_q$ with $q = 256$ and dimension 32. On the other hand, the internal code is a first-order Reed-Muller code $\mathcal{R}(1, m)$ with $m = 7$. Hence, the Reed-Muller code is a [128,8,64] binary code (over $\mathbb{F}_2$). Specifically, the authors of HQC chose to use shortened Reed-Solomon codes and duplicated Reed-Muller codes. The shortening operation has been extensively covered and, therefore, does not warrant further discussion in this context. Table 5.4 displays the original Reed-Solomon codes along with their corresponding shortened versions.

On the other hand, regarding the duplication operation, it is essential to highlight that it is equivalent to introducing an additional repetition encoder downstream of the RM encoder with a length precisely matching the duplication factor. In particular, duplicating the RM code involves repeating each symbol output from the RM encoder a number of times equal to the duplication factor. Therefore, to be more explicit, assuming d is the duplication factor, equation (5.9) becomes:

$$\mathbf{c}'' = \left(c''^{(0)}_{0,0} \ldots c''^{(0)}_{0,d-1} \ldots c''^{(0)}_{n_i-1,0} \ldots c''^{(0)}_{n_i-1,d-1} \ldots c''^{(n_e-1)}_{0,0} \ldots c''^{(n_e-1)}_{0,d-1} \ldots c''^{(n_e-1)}_{n_i-1,0} \ldots c''^{(n_e-1)}_{n_i-1,d-1}\right) \quad (5.10)$$

SL	Reed-Muller code	Duplication	Duplicated Reed-Muller Code
128	[128,8,64]	2	[256,8,128]
192	[128,8,64]	4	[512,8,256]
256	[128,8,64]	6	[768,8,384]

Table 5.5: Duplicated Reed-Muller codes.

SL	n_1	n_2	n	w	w_r	w_e
128	80	256	20,533	67	77	77
192	76	512	38,923	101	117	117
256	78	768	59,957	133	153	153

Table 5.6: HQC Parameters for the Round 2 submission using concatenated codes

Table 5.5 shows different parameter sets for duplicated Reed-Muller codes. Table 5.6 shows the parameter sets for HQC assuming a concatenated code is employed. Specifically, the concatenated code used consists of a $[n_2,\ 8,\ n_2/2]$ Reed-Muller code as inner code and a $[n_1, 32, n_1 - k + 1]$ Reed-Solomon code as the outer code.

5.3.3 3rd and 4th round submission

In the context of the third and fourth NIST rounds, there are no distinctions in the codes employed or the parameters. However, in the fourth NIST round, the HQC authors made small adjustments to enhance security. Specifically, they introduced a public salt to mitigate multi-ciphertext attacks and enhanced the hardware implementation to protect against side-channel attacks based on the evaluation of the execution time.

Notably, in the third and fourth round of the NIST submission, the HQC authors made a strategic decision to employ a concatenated code structure. This structure involves the utilization of an external code, specifically a Reed-Solomon code over the finite field $\mathbb{F}_q$ with q set to 256. Explicitly, depending on the different SL required, shortened versions were used. In particular, shortened Reed-Solomon codes denoted by RS-S1, RS-S2, and RS-S3 are obtained

Code	n	k	t_{RS}
RS-1	255	225	15
RS-2	255	223	16
RS-3	255	197	29
RS-S1	46	16	15
RS-S2	56	24	16
RS-S3	90	32	29

Table 5.7: Original and Shortened RS codes

SL	Reed-Muller code	Duplication	Duplicated Reed-Muller Code
128	[128,8,64]	3	[384,8,192]
192	[128,8,64]	5	[640,8,320]
256	[128,8,64]	5	[640,8,320]

Table 5.8: Duplicated Reed-Muller codes.

from RS-1, RS-2, and RS-3, respectively. To provide a comprehensive overview of the parameters associated with both the original Reed-Solomon codes and their resulting shortened versions, we turn to Table 5.7.

Conversely, the inner code was chosen to be a first-order Reed-Muller code, specifically identified as $\mathcal{R}(1, m)$, with m set to 7. This Reed-Muller code, operating in the binary domain, exhibits parameters $[128, 8, 64]$. To achieve the desired code configurations, the Reed-Muller code will undergo a duplication process, occurring either three or five times, resulting in the codes delineated in Table 5.8.

The recommended parameters for this version of HQC are shown in Table 5.9. Specifically, n_1 indicates the length of the Reed-Solomon code while n_2 the length of the Reed-Muller code, therefore, $n_1 n_2$ is the length of the concatenated code (the ambient space has lenght n, the smallest primitive prime greater than $n_1 n_2$ to avoid algebraic attacks).

SL	n_1	n_2	n	w	w_r	w_e
128	46	384	17,669	66	75	75
192	56	640	35,851	100	114	114
256	90	640	57,637	131	149	149

Table 5.9: HQC Parameters for Round 3 and 4 submission

The HQC version introduced in the second NIST round represents a transitional phase, marking a shift from using a tensor product code (BCH+REP) to a concatenated code structure involving non-binary Reed-Solomon and binary Reed-Muller codes. This is why the first and third versions of HQC are often referenced in subsequent discussions as they signify crucial stages of innovation. For the sake of maintaining generality and clarity throughout the ongoing discourse in this thesis, we shall consistently employ the terms "internal code" (alternatively referred to as "inner code") to represent the REP or Reed-Muller code. In parallel, we will employ the terms "external code" (or "outer code") to denote the BCH or Reed-Solomon code. This standardized terminology will facilitate a more comprehensive understanding of the coding mechanisms employed within the context of HQC and its subsequent description.

5.4 Representation of objects

Elements of $\mathbb{F}_2^n$, $\mathbb{F}_2^{n_1 n_2}$ and $\mathbb{F}_2^k$ are represented as binary arrays. Furthermore, with the goal of reducing the size of objects, a seed expander is utilized. This seed expander is based on SHAKE256, a cryptographic hash function that belongs to the SHA-3 family. To clarify, seed expansion is the process of taking a relatively short and random seed or initial value and transforming it into a longer and more unpredictable sequence of bits. SHAKE256 is well-suited for this purpose because it can generate an arbitrary amount of pseudorandom data with variable length. In essence, this capability is achieved through the sponge construction on which SHA-3 hash functions are built, enabling native implementation of an Extendable-Output Function (XOF) to adjust the output length as needed. To be more specific, SHAKE256 takes an initial seed as input and generates a stream of bits that exhibit a high degree of randomness. This feature makes it valuable for various applications, including key genera-

	pk	sk	ciphertext	ss
size [bit]	$320 + n$	320	$512 + 2n$	512
size [byte]	$40 + \lceil \frac{n}{8} \rceil$	40	$64 + \lceil \frac{n}{4} \rceil$	64

Table 5.10: Resulting size for HQC using NIST seed expander initialized with 40 bytes long seeds.

tion, nonce generation, and the generation of random values for cryptographic protocols. In this particular case, SHAKE256 is initialized with a 40-byte string, which serves as the seed. The secret key $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$ is represented as $\mathsf{sk} = (\mathbf{seed1})$ where **seed1** is used to generate $\mathbf{x}$ and $\mathbf{y}$. The public key $\mathsf{pk} = (\mathbf{h}, \mathbf{s})$ is represented as $\mathsf{pk} = (\mathbf{seed2}, \mathbf{s})$ where **seed2** is used to generate $\mathbf{h}$. The ciphertext $\mathbf{c}$ is represented as $(\mathbf{u}, \mathbf{v}, \mathbf{d})$ where $\mathbf{d}$ is generated using SHAKE256-512. Therefore, the secret key has size 320 (in bits), the public key has size $320 + n$ (in bits) and the ciphertext has size $2n + 512$ (in bits). The shared key, referred to as ss, has a size of 512 bits, corresponding to the output size of SHA-512. To summarize, refer to Table 5.10.

5.5 Estimating the DFR for HQC

In this section we recall how the DFR analysis for HQC proceeds. We report, with full details, an analysis which is analogous to the one in the HQC submission, but generalize it to the case in which the polynomials $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ have different weights, which we indicate respectively by $w_r^{(1)}$ and $w_r^{(2)}$. We do this to make the thesis self contained and, more importantly, to prepare the stage for our new decoder, which we introduce in the next chapter. Indeed, we anticipate that our decoding strategy starts by guessing, with a correlation-based approach, some of the set coefficients in the polynomials $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$. This allows to reduce the noise term superimposed on $\mathbf{mG}$ by an entity that depends on the number of coefficients we have correctly guessed for each polynomial. As we detail in the next chapter, when the number of guessed coefficients for

$\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ is not the same, the DFR analysis is derived from the standard analysis of HQC, with the only precaution that one must take into account different weights for such polynomials. We start by considering the following preliminary result:

Proposition 1. *For two polynomials* $\mathbf{a} \in \mathcal{R}(w_a)$ *and* $\mathbf{b} \xleftarrow{\$} \mathcal{R}(w_b)$*, a coefficient (say, the first one) in the product* $\mathbf{a} \cdot \mathbf{b}$ *is set with probability*

$$\rho_{w_a,w_b} = \sum_{\substack{\ell \in [1;\min\{w_a,w_b\}] \\ \ell \text{ odd}}} \frac{\binom{w_a}{\ell}\binom{n-w_a}{w_b-\ell}}{\binom{n}{w_b}}. \tag{5.11}$$

Proof. According to the rules of polynomial multiplication, we have:

$$c_k = \sum_{i+j \equiv k \bmod n} a_i \cdot b_j, \quad \text{for } k \in \{0, 1, \cdots n-1\}.$$

where $c_k = 1$ only when it constitutes the sum of an odd quantity of '1s'. Consequently, out of the $\binom{n}{w_a}\binom{n}{w_b}$ potential combinations of $\mathbf{a}$ and $\mathbf{b}$, we need to focus on those where $a_i \cdot b_j = 1$ an odd number of times. Let C_ℓ represent the count of pairs $(\mathbf{a}, \mathbf{b})$ where $a_i \cdot b_j = 1$ exactly ℓ times. When $\ell > \min(w_a, w_b)$, it is evident that $C_\ell = 0$. Indeed, we are evaluating the various ways in which ℓ positions out of the total n can be chosen such that both vectors $\mathbf{a}$ and $\mathbf{b}$ have corresponding elements set to 1. Since $\ell > \min(w_a, w_b)$, there is no pair $(\mathbf{a}, \mathbf{b})$ for which $a_i \cdot b_j = 1$ exactly ℓ times. On the other hand, when $\ell \leqslant \min(w_a, w_b)$, we are assessing the various ways in which ℓ positions out of the total n can be selected so that the elements set to 1 in both $\mathbf{a}$ and $\mathbf{b}$ coincide in those ℓ positions. Hence, we can start by choosing ℓ positions out of the total n, resulting in $\binom{n}{\ell}$ different possibilities. For each choice of the ℓ positions, we must select $w_a - \ell$ elements from $\mathbf{a}$ and $w_b - \ell$ elements from $\mathbf{b}$, avoiding any 'intersection' with the first ℓ elements. There are $\binom{n-\ell}{w_a-\ell}$ ways to select such that elements from $\mathbf{a}$ and $\binom{n-w_a}{w_b-\ell}$ ways to select such that elements from $\mathbf{b}$.

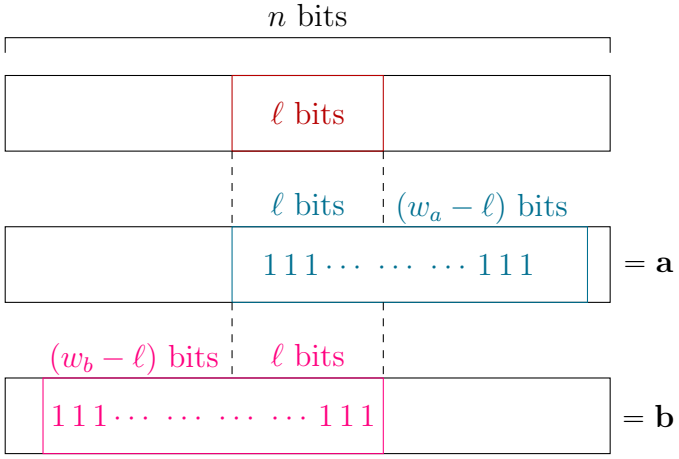

Figure 5.2: Graphical representation of **a** and **b**. The ones do not need to be consecutive. This assumption was made only for mere graphical convenience.

Therefore, when $\ell \leqslant \min(w_a, w_b)$ it follows that:

$$C_\ell = \binom{n}{\ell}\binom{n-\ell}{w_a-\ell}\binom{n-w_a}{w_b-\ell}. \tag{5.12}$$

From a visual perspective, let's refer to Figure 5.2. Hence:

$$P(c_k = 1) \triangleq \rho_{w_a,w_b} = \frac{1}{\binom{n}{w_a}\binom{n}{w_b}} \sum_{\substack{1\leqslant\ell\leqslant\min(w_a,w_b)\\ \ell \text{ odd}}} C_\ell \tag{5.13}$$

By substituting equation (5.12) in (5.13), we obtain:

$$\rho_{w_a,w_b} = \frac{1}{\binom{n}{w_a}\binom{n}{w_b}} \sum_{\substack{1\leqslant\ell\leqslant\min(w_a,w_b)\\ \ell \text{ odd}}} \binom{n}{\ell}\binom{n-\ell}{w_a-\ell}\binom{n-w_a}{w_b-\ell} \tag{5.14}$$

At this point, let's focus our attention on the term:

$$\frac{\binom{n}{\ell}\binom{n-\ell}{w_a-\ell}}{\binom{n}{w_a}} = \frac{\frac{n!}{\ell!(n-\ell)!}\cdot\frac{(n-\ell)!}{(w_a-\ell)![n-\ell-(w_a-\ell)]!}}{\frac{n!}{w_a!(n-w_a)!}} = \frac{w_a!}{\ell!(w_a-\ell)!} = \binom{w_a}{\ell} \tag{5.15}$$

Ultimately, upon substituting the outcomes derived from equation (5.15) into equation (5.14), we successfully arrive at our intended proof.

□

In our examination of the product's total weight, we incorporate a foundational assumption that is not only central to our study but also finds its application in the official HQC specification that has been submitted.

Assumption 2. *Let* $\mathbf{a} \in \mathcal{R}(w_a)$ *and* $\mathbf{b} \xleftarrow{\$} \mathcal{R}(w_b)$. *Then, we assume that all the coefficients in their product are independent and uncorrelated random variables, following a Bernoulli distribution with parameter* ρ_{w_a,w_b} *as in* (5.11).

Employing the above assumption, one has the following Proposition, which establishes the weight distribution of the polynomial $\mathbf{z}$.

Proposition 2. *Let* $\mathbf{x}, \mathbf{y} \in \mathcal{R}(w)$, $\mathbf{r}^{(1)} \xleftarrow{\$} \mathcal{R}(w_r^{(1)})$, $\mathbf{r}^{(2)} \xleftarrow{\$} \mathcal{R}(w_r^{(2)})$, $\rho_e = \frac{w_e}{n}$ *and* $\mathbf{e} \sim \mathscr{B}_{n,\rho_e}$. *Under assumption 2, the polynomial* $\mathbf{z} = \mathbf{x} \cdot \mathbf{r}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{e}$ *follows a Bernoulli distribution with parameter*

$$\rho_z = 4\rho_{w,w_r^{(1)}}\rho_{w,w_r^{(2)}}\rho_e + \rho_{w,w_r^{(1)}} + \rho_{w,w_r^{(2)}} + \rho_e - 2\left(\rho_{w,w_r^{(1)}}\rho_{w,w_r^{(2)}} + \rho_{w,w_r^{(1)}}\rho_e + \rho_{w,w_r^{(2)}}\rho_e\right)$$

Proof. We consider that each product $\widehat{\mathbf{x}} = \mathbf{x} \cdot \mathbf{r}^{(2)}$, $\widehat{\mathbf{y}} = \mathbf{y} \cdot \mathbf{r}^{(1)}$ is a Bernoulli distributed vector with respective parameter $\rho_{w,w_r^{(2)}}$ and $\rho_{w,w_r^{(1)}}$. Let us focus on a single coefficient of $\mathbf{z}$ (say, the i-th one): it will be set with probability

$$P(\widehat{x}_i = 1) \cdot P(\widehat{y}_i = 0) \cdot P(e_i = 0) + P(\widehat{x}_i = 0) \cdot P(\widehat{y}_i = 1) \cdot P(e_i = 0) + P(\widehat{x}_i = 0) \cdot P(\widehat{y}_i = 0) \cdot P(e_i = 1) + P(\widehat{x}_i = 1) \cdot P(\widehat{y}_i = 1) \cdot P(e_i = 1).$$

Substituting the probabilities with the associated Bernoulli parameters we get:

$$\rho_z = (1 - \rho_{w,w_r^{(1)}})(1 - \rho_{w,w_r^{(2)}})\rho_e + (1 - \rho_{w,w_r^{(1)}})\rho_{w,w_r^{(2)}}(1 - \rho_e) + \rho_{w,w_r^{(1)}}(1 - \rho_{w,w_r^{(2)}})(1 - \rho_e) + \rho_{w,w_r^{(1)}}\rho_{w,w_r^{(2)}}\rho_e$$

After some manipulations, we obtain the expression for ρ_z. □

Notice that, to be formally correct, the parameter ρ_z should be deemed

as a function of $w_r^{(1)}$ and $w_r^{(2)}$; yet, to avoid burdening the notation, we do not explicitly indicate such a dependence. Now, we are ready to evaluate the DFR of the HQC cryptosystem. In the analysis, another assumption is needed, which we formalize as follows.

Assumption 3. *Let* $\mathbf{x}, \mathbf{y} \in \mathcal{R}(w)$, $\mathbf{r}^{(1)} \xleftarrow{\$} \mathcal{R}(w_r^{(1)})$, $\mathbf{r}^{(2)} \xleftarrow{\$} \mathcal{R}(w_r^{(2)})$ *and* $\mathbf{e} \xleftarrow{\$} \mathcal{R}(w_e)$. *We assume that distribution of the weight of* $\mathbf{z} = \mathbf{x}\cdot\mathbf{r}^{(2)} + \mathbf{y}\cdot\mathbf{r}^{(1)} + \mathbf{e}$, *for what concerns the DFR of the algorithm* $\mathfrak{D}$, *is essentially the same we would obtain when* $\mathbf{z} \sim \mathscr{B}_{n,\frac{w_z}{n}}$.

Proposition 3. *Under Assumptions 2 and 3, the DFR of HQC is effectively approximated, independently of the specific NIST round being considered, by:*

$$\epsilon(w_r^{(1)}, w_r^{(2)}) = \sum_{t=0}^{w\left(w_r^{(1)}+w_r^{(2)}\right)+w_e} \underbrace{f_{n,\rho_z}(t)}_{P(\mathbf{z}\ has\ weight\ t)} \cdot \underbrace{\left(1 - \sum_{i=0}^{t_{outer}} f_{n_i,\rho_{inner}(t)}(i)\right)}_{P(no.\ of\ inner\ errors > t_{outer}\,|\,\mathbf{z}\ has\ weight\ t)}$$

Proof. From theorems (4.2) and (4.3), we get that the maximum Hamming weight of $\mathbf{z}$ (neglecting hypothetical cancellations) is $w \cdot (w_r^{(1)} + w_r^{(2)}) + w_e$. Precisely, $\mathbf{z}$ has weight $t \in \left[0;\ w(w_r^{(1)} + w_r^{(2)}) + w_e\right]$ with probability given by:

$$f_{n,\rho_z}(t) = \binom{n}{t} \rho_z{}^t (1-\rho_z)^{n-t}$$

where ρ_z is defined in Proposition 2.

According to Assumption 3, we assume that $\mathbf{z}$ is distributed according to a Bernoulli distribution with parameter t/n. We now focus on one codeword of the inner code (REP or RM). The likelihood of inner decoding failure is equal to the probability of $\mathbf{z}$ having a weight exceeding the error correction capacity of the inner code, i.e., $t_{\text{inner}} = \left\lceil \frac{d_i - 1}{2} \right\rceil$, where d_i is the minimum distance of the inner code. As a result, the probability of inner decoding failure when

assuming $\mathrm{wt}(\mathbf{z}) = t$ can be expressed as follows:

$$\rho_{\text{inner}}(t) = \sum_{i=t_{\text{inner}}+1}^{n_i} f_{n_i,\,\rho_z}(i).$$

External decoding fails when the remaining errors after internal decoding exceed the external code's error correction capability (BCH or RS). This occurs with probability:

$$1 - \sum_{i=0}^{t_{\text{outer}}} f_{n_i,\,\rho_{\text{inner}}(t)}(i).$$

Finally, by taking into account all possible weights for $\mathbf{z}$, we obtain the DFR of HQC, denoted by $\epsilon(w_r^{(1)}, w_r^{(2)})$, as expressed in Proposition 3. □

Chapter 6

New Decoder for HQC

6.1 Idea leading to the new HQC decoder

The consideration that leads us to the concept of the new HQC decoder is closely tied to information theory. Specifically, there is information that is not used in the decoding process, and by harnessing it, HQC decoding can be made more efficient. Explicitly, in HQC the decoding algorithm $\mathfrak{D}$ is applied on input $\mathbf{c} = \mathbf{v} + \mathbf{y} \cdot \mathbf{u} = \mathbf{mG} + \mathbf{z}$, where $\mathbf{z} = \mathbf{x} \cdot \mathbf{r}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{e}$ is assumed to be random-like. However, in reality, $\mathbf{z}$ is not entirely random because the legitimate user knows the secret key $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$. Therefore, this information can be used to enhance HQC decoding. In this section, we delineate the new decoding method we put forth for HQC. We present the corresponding procedure, which takes as input $\mathbf{c} \in \mathcal{R}$ and an integer threshold $\beta \in [0; w]$ in Figure 6.1.

Basically, the algorithm first decodes through the inner code and then re-encodes through the inner code itself in order to get a first (rough) estimate on the error vector $\mathbf{z}$ affecting the codeword $\mathbf{mG}$. This estimate, denoted by $\widehat{\mathbf{e}}$, is employed to guess the positions of set coefficients for the polynomials $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$. To do this, the algorithm first computes the correlation between the obtained estimate $\widehat{\mathbf{e}}$ and shifted versions of the secret key polynomials.

Decoder $\mathfrak{D}$ ($\mathbf{c} \in \mathcal{R}, \mathsf{sk} := (\mathbf{x}, \mathbf{y}) \in \mathcal{R}^2$):

1) for $\alpha = 0, \cdots, n_i - 1$ do:

   1.1) set $\big(b_{\alpha \cdot k_i}, \cdots, b_{(\alpha+1)\cdot k_i - 1}\big) = \text{DecInner}\big(c_{\alpha \cdot n_e}, \cdots, c_{(\alpha+1)\cdot n_e - 1}\big)$;

   1.2) set $\big(a_{\alpha \cdot n_e}, \cdots, a_{(\alpha+1)\cdot n_e - 1}\big) = \text{EncInner}\big(b_{\alpha \cdot k_i}, \cdots, b_{(\alpha+1)\cdot k_i - 1}\big)$;

   1.3) set $\widehat{e}_{\alpha \cdot n_e + j} = 1$ for every $j \in [0; n_e - 1]$ such that $c_{\alpha \cdot n_e + j} \neq a_{\alpha \cdot n_e + j}$

2) set $\mathbf{s}^{\mathbf{x}} = \text{Correlation}(\widehat{\mathbf{e}}, \mathbf{x})$ and $\mathbf{s}^{\mathbf{y}} = \text{Correlation}(\widehat{\mathbf{e}}, \mathbf{y})$;

3) define polynomials $\widehat{\mathbf{r}}^{(1)}, \widehat{\mathbf{r}}^{(2)} \in \mathcal{R}$, so that

$$\widehat{r}_i^{(1)} = \begin{cases} 1 & \text{if } s_i^{\mathbf{y}} \geqslant \beta, \\ 0 & \text{otherwise,} \end{cases}$$

$$\widehat{r}_i^{(2)} = \begin{cases} 1 & \text{if } s_i^{\mathbf{x}} \geqslant \beta, \\ 0 & \text{otherwise;} \end{cases}$$

4) set $\widehat{\mathbf{z}} = \mathbf{x} \cdot \widehat{\mathbf{r}}^{(2)} + \mathbf{y} \cdot \widehat{\mathbf{r}}^{(1)}$;

5) set $\Delta\mathbf{c} = \mathbf{c} + \widehat{\mathbf{z}}$;

6) decode $\Delta\mathbf{c}$

Figure 6.1: New decoding algorithm for HQC. In this figure n_e represents the external code's length (which depends on the HQC version, either BCH or Reed Solomon code), while n_i stands for the inner code's length (which varies with the HQC version, either REP or Reed Muller code).

Namely, focusing on $\mathbf{x}$, we have that the i-th entry of $\mathbf{s}^{\mathbf{x}}$ is obtained as

$$s_i^{\mathbf{x}} = \sum_{j \in \{\ell + i \mod n \mid \ell \in \text{supp}(\mathbf{x})\}} \mathfrak{L}(\widehat{e}_j),$$

where $\mathfrak{L}$ denotes the lifting operation from $\mathbb{F}_2$ to $\mathbb{N}$. It is easy to see that $s_i^{\mathbf{x}}$ takes values in $[0; w]$ and is expected to have large values when $r_i^{(2)} = 1$. Indeed, let us assume for the moment that $\widehat{\mathbf{e}} = \mathbf{z}$. Let $r_i^{(2)} = 1$ and $\widetilde{\mathbf{r}}^{(2)}$ be the polynomial obtained from $\mathbf{r}^{(2)}$ by setting the i-th coefficient to 0. Then, the

following holds

$$s_i^{\mathbf{x}} = \sum_{j \in \mathrm{supp}(\mathbf{x})} \mathfrak{L}\left(1 + \left(\mathbf{x} \cdot \widetilde{\mathbf{r}}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{e}\right)_{j+i \mod n}\right). \tag{6.1}$$

Since all the involved polynomials are sparse, we expect that all the coefficients $\left(\mathbf{x} \cdot \widetilde{\mathbf{r}}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{e}\right)_{j+i \mod n}$ are equal to 0 with large probability (say, significantly larger than 1/2). The correlation is obtained by summing w variables that are set to 1 with large probability, hence it is expected to be large (say, higher than $w/2$). When, instead, $r_i^{(2)} = 0$, we have

$$s_i^{\mathbf{x}} = \sum_{j \in \mathrm{supp}(\mathbf{x})} \mathfrak{L}\left(\left(\mathbf{x} \cdot \mathbf{r}^{(2)} + \mathbf{y} \cdot \mathbf{r}^{(1)} + \mathbf{e}\right)_{j+i \mod n}\right). \tag{6.2}$$

For analogous reasons (i.e., summing coefficients of a sparse polynomial), in this case we expect to have a correlation value which is rather low (say, lower than $w/2$). Clearly, the decoder operates without full knowledge of the exact error vector $\mathbf{z}$; instead, it relies on a preliminary estimate acquired through the initial decoding and re-encoding steps performed in accordance with the inner code. So, it becomes necessary to adapt the expressions in (6.1) by considering the coefficients of $\widehat{\mathbf{e}}$. However, in cases where the inner code is performing efficiently, successfully decoding a substantial number of errors, it is reasonable to anticipate that $\widehat{\mathbf{e}}$ and $\mathbf{z}$ will share a considerable number of identical coefficients. This observation reinforces the validity of the previous explanation. An analogous reasoning holds for the coefficients of $\mathbf{r}^{(1)}$.

In the end, when the value of β is properly chosen, we expect $\widehat{\mathbf{r}}^{(1)}$ and $\widehat{\mathbf{r}}^{(2)}$ to be good approximations of $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$, respectively. In other words, we expect the polynomials $\Delta\mathbf{r}^{(i)} = \mathbf{r}^{(i)} + \widehat{\mathbf{r}}^{(i)}$, for $i \in \{1, 2\}$, to have a weight which is lower than w_r. These new polynomials can be used to remove some of the noise affecting the codeword. Indeed, it is easy to see that

$$\Delta\mathbf{c} = \mathbf{mG} + \Delta\mathbf{z},$$

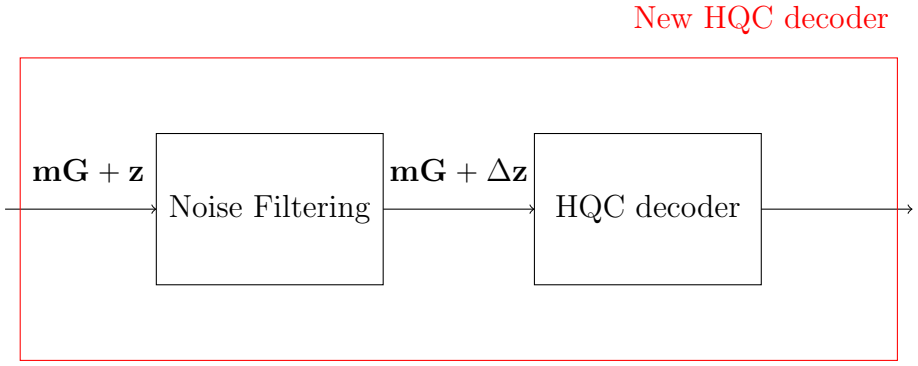

Figure 6.2: Link between the classic HQC decoder and the new proposal. It's worth noticing that the difference lies in the initial noise filtering step.

with

$$\Delta \mathbf{z} = \mathbf{x} \cdot \Delta \mathbf{r}^{(2)} + \mathbf{y} \cdot \Delta \mathbf{r}^{(1)} + \mathbf{e}.$$

Since we expect $\mathrm{wt}(\Delta \mathbf{r}^{(i)}) < w_r$, it holds that, with very high probability, $\mathrm{wt}(\Delta \mathbf{z}) < \mathrm{wt}(\mathbf{z})$. Decoding of $\Delta \mathbf{c}$ should fail at a lower probability. Essentially, our proposed decoder can be thought of as the classical HQC decoder, with an additional filtering stage in which the noise entity is reduced as illustrated in Figure 6.2.

6.2 Complexity of the new Decoder

In the preceding section, we observed that our decoder can be viewed as the conventional HQC decoder complemented by an extra denoising phase. One noteworthy characteristic of this added step is its notably low computational complexity. In this regard, we present the following Proposition:

Proposition 4. *The running time of the proposed decoder is*

$$O\left(T_{\text{Inner}} + T_{\text{Outer}} + n + nw + ww_r\right).$$

Proof. First, we need to distinguish between two specific scenarios according to the HQC version being considered. The first scenario involves the use of a repetition code as the internal code, while the second scenario pertains to the

external code being a first-order Reed-Muller code. In the former case, both the encoding and decoding costs are characterized by $O(n)$. However, in the latter case, when we employ the Fast Walsh-Hadamard Transform (FWHT) for both encoding and decoding, it results in a computational cost of $O(n \log_2 n)$. To provide further clarity and maintain a general perspective, we designate T_{Inner} as the cost associated with encoding or decoding the internal code and T_{Outer} as the cost related to decoding the external code. After this necessary clarification, we can proceed with the evaluation of the computational cost of the new decoder. We first decode through the inner code, with cost T_{Inner} and then re-encode through the inner code itself with cost T_{Inner} in order to get a rough estimate of $\mathbf{z}$.Then we set the polynomial $\widehat{e}$ with cost $O(n)$. Computing the correlation for a polynomial and a position i comes with cost $O(w)$, since we just need to sum w coefficients. Then, the overall correlation computation takes time $O(nw)$ for both $\mathbf{x}$ and $\mathbf{y}$. When the value of the threshold β is properly chosen, we can safely consider that the weights of the polynomial $\widehat{\mathbf{r}}^{(i)}$ is never larger than w_r, so that to compute $\widehat{\mathbf{z}}$ we can exploit the sparse nature of the involved polynomials and hence have a cost of $O(ww_r)$. Analogously, we can compute $\Delta\mathbf{z}$ with cost $O(ww_r)$. We finally apply the classical HQC decoding procedure, with cost $T_{\text{Inner}} + T_{\text{Outer}}$. □

Remark 1. *The increase in the complexity of the decryption phase is rather limited. For instance, in HQC with a repetition code, we have* $T_{\text{Inner}} = O(n)$. *Hence, roughly, the overall increase corresponds to* $3n + 2nw + 2ww_r$, *which is linear in* n. *On the other hand, in HQC with a first-order Reed-Muller code, we have* $T_{\text{Inner}} = O(n \log_2 n)$. *Therefore, in this case, the overall increase corresponds to* $n + 2\log_2 n + 2nw + 2ww_r$, *which is linear in* n.

Clearly, given the limited increase in complexity associated with the introduction of the initial noise filtering step, it is also possible to introduce an iterative decoding such that instead of a single filtering step there are multiple steps.

6.3 DFR of the new Decoder

In this section, we will discuss the analysis of the DFR for the new decoder. To begin, our initial focus is on assessing the weight distribution of $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$. Specifically, for $i \in \{1, 2\}$, let $w_r^{(i)} = \mathrm{wt}(\Delta\mathbf{r}^{(i)})$. According to the law of total probability, the probability of having a decoding failure (an event which we denote with $\perp$) can be written as

$$\epsilon(n_1, n_2) = \sum_{w_r^{(1)}, w_r^{(2)}} P\big(\perp \mid w_r^{(1)}, w_r^{(2)}\big) \cdot P\big(w_r^{(1)}, w_r^{(2)}\big),$$

where $P\Big(\perp \mid w_r^{(1)}, w_r^{(2)}\Big)$ is the probability to have a decoding failure given that $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$ have weights $w_r^{(1)}$ and $w_r^{(2)}$, respectively, while $P\Big(w_r^{(1)}, w_r^{(2)}\Big)$ is the probability that these two polynomials assume such weights. To simplify the analysis, we assume independence of the weights of $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$, i.e.,

$$P\big(w_r^{(1)}, w_r^{(2)}\big) \approx P\big(w_r^{(1)}\big) \cdot P\big(w_r^{(2)}\big).$$

Notice that $\Delta\mathbf{r}_1$ and $\Delta\mathbf{r}_2$ follow the same probability distribution, so we can focus only on one of them, say, $\Delta\mathbf{r}^{(2)}$. Its weight distribution is derived in the following propositions.

Proposition 5. ***Probability to wrongly guess a 1-coefficient***
Let j such that $r_j^{(i)} = 1$. Then, the probability that $\widehat{r}_j^{(i)} = 0$ is defined as

$$\tau_{1\to 0} = \sum_{\ell=0}^{\beta-1} \binom{w}{\ell} \widehat{\rho}^{\,\ell} (1-\widehat{\rho})^{w-\ell},$$

where

$$\widehat{\rho} = (1-\widetilde{\rho}) \sum_{\ell=0}^{t-1} \binom{n_i-1}{\ell} \rho_z^{\ell} (1-\rho_z)^{n_i-1-\ell} + \widetilde{\rho} \sum_{\ell=t}^{n_i-1} \binom{n_i-1}{\ell} \rho_z^{\ell} (1-\rho_z)^{n_i-1-\ell},$$

with d_i, n_i and $t = \lfloor \frac{d_i-1}{2} \rfloor$ denoting the minimum distance, the length and the

error correcting capability of the inner code, respectively, and

$$\widetilde{\rho} = \widetilde{\rho}_{w,w_r-1}\left(1-\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)-(1-\rho_{w,w_r})\frac{w_e}{n}\right) + (1-\widetilde{\rho}_{w,w_r-1})\left(\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)+(1-\rho_{w,w_r})\frac{w_e}{n}\right),$$

$$\widetilde{\rho}_{w,w_r-1} = \sum_{\substack{\ell\in[1;\min\{w,w_r-1\}]\\ \ell \text{ odd}}} \frac{\binom{w}{\ell}\binom{n-w}{w_r-1-\ell}}{\binom{n}{w_r-1}}.$$

Proof. We consider an index $i \in \mathrm{supp}(\mathbf{r}^{(2)})$, and derive the probability distribution of the correlation value. Let $\widetilde{\mathbf{r}}^{(2)}$ be the polynomial whose coefficients are defined as follows

$$\widetilde{r}_\ell^{(2)} = \begin{cases} 0 & \text{if } \ell = i, \\ r_\ell^{(2)} & \text{otherwise.} \end{cases}$$

Observe that $\mathrm{wt}(\widetilde{\mathbf{r}}^{(2)}) = w_r - 1$ due to $r_i^{(2)} = 1$. The correlation is obtained by summing the coefficients of the estimated error vector $\widehat{\mathbf{e}}$ in the positions indexed by $j \in \mathrm{supp}(\mathbf{x}) + i = \{\ell + i \mod n \mid \ell \in \mathrm{supp}(\mathbf{x})\}$. Let us first consider the corresponding coefficient in $\mathbf{z}$; notice that

$$z_j = 1 + (\mathbf{x}\cdot\widetilde{\mathbf{r}}^{(2)})_j + (\mathbf{y}\cdot\mathbf{r}^{(\mathbf{1})} + \mathbf{e})_j = 1 + \widetilde{z}_j,$$

where $\widetilde{\mathbf{z}} = \mathbf{x}\cdot\widetilde{\mathbf{r}}^{(2)} + \mathbf{y}\cdot\mathbf{r}^{(\mathbf{1})} + \mathbf{e}$. With arguments analogous to those in Proposition 2, it is straightforward to see that $(\mathbf{y}\cdot\mathbf{r}^{(\mathbf{1})} + \mathbf{e})_j$ is Bernoulli distributed with parameter

$$\rho_{w,w_r}\left(1-\frac{w_e}{n}\right) + (1-\rho_{w,w_r})\frac{w_e}{n}.$$

Moreover, also $(\mathbf{x}\cdot\widetilde{\mathbf{r}}^{(2)})_j$ is Bernoulli distributed, with parameter

$$\widetilde{\rho}_{w,w_r-1} = \sum_{\substack{\ell\in[1;\min\{w,w_r-1\}]\\ \ell \text{ odd}}} \frac{\binom{w-1}{\ell}\binom{n-w}{w_r-1-\ell}}{\binom{n}{w_r-1}}.$$

Hence, $\widetilde{z}_j$ is equal to 1 with probability

$$\widetilde{\rho} = \widetilde{\rho}_{w,w_r-1}\left(1-\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)-(1-\rho_{w,w_r})\frac{w_e}{n}\right) + (1-\widetilde{\rho}_{w,w_r-1})\left(\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)+(1-\rho_{w,w_r})\frac{w_e}{n}\right)$$

Let $\perp_{\text{Inner}}(j)$ indicate the event that the inner code codeword in which the j-th coordinate is contained is wrongly decoded. The complementary event (i.e., a decoding success) is indicated as $\overline{\perp}_{\text{Inner}}(j)$. Then, we have

$$P(\widehat{e}_j = 1) = P\big(\overline{\perp}_{\text{Inner}}(j) \mid \widetilde{z}_j = 0\big)\cdot P(\widetilde{z}_j = 0) + P(\perp_{\text{Inner}}(j) \mid \widetilde{z}_j = 1)P(\widetilde{z}_j = 1). \tag{6.3}$$

Let $\text{supp}_{\text{inner}}(j)$ denote the set of indices which correspond to the same inner codeword as position j. Then, $\widehat{e}_j = 1 = z_j = 1$ if and only if the remaining n_i positions allow correct decoding of position j, despite position j being erroneous. For a generic inner code with minimum distance d_i, it is straightforward to conclude that this happens whenever the number of set coefficients in $\text{supp}\,\text{inner}(j)\backslash\{j\}$ is not larger than $t-1 = \lfloor\frac{d_i-1}{2}\rfloor - 1$. The analysis given by standard HQC implies that it is justified to assume that the positions in $\text{supp}_{\text{inner}}(j)\backslash\{j\}$ are independently Bernoulli distributed with parameter ρ_z as in Proposition 2. Hence, the required probabilities can be calculated as

$$P\big(\overline{\perp}_{\text{Inner}}(j) \mid \widetilde{z}_j = 0\big) = \sum_{\ell=0}^{t-1}\binom{n_i-1}{\ell}\rho_z^\ell(1-\rho_z)^{n_i-1-\ell}.$$

With analogous reasoning, we obtain

$$P(\perp_{\text{Inner}}(j) \mid \widetilde{z}_j = 1) = \sum_{\ell=t}^{n_i-1}\binom{n_i-1}{\ell}\rho_z^\ell(1-\rho_z)^{n_i-1-\ell}.$$

Then, indicating $\widehat{\rho} = P(\widehat{e}_j = 1)$, we can rewrite (6.3) as

$$\widehat{\rho} = (1-\widetilde{\rho})\sum_{\ell=0}^{t-1}\binom{n_i-1}{\ell}\rho_z^\ell(1-\rho_z)^{n_i-1-\ell} + \widetilde{\rho}\sum_{\ell=t}^{n_i-1}\binom{n_i-1}{\ell}\rho_z^\ell(1-\rho_z)^{n_i-1-\ell} \tag{6.4}$$

We model $\sum_{\text{supp}\,\mathbf{x}+i}\widehat{e}_j$ as a sum of independent random variables. Then, $\sum_{\text{supp}\,\mathbf{x}+i}\widehat{e}_j$ follows the binomial distribution with w trials and success probability $\widehat{\rho}$. Consequently, we obtain

$$\tau_{1\to 0}=\sum_{\ell=0}^{\beta-1}\binom{w}{\ell}\widehat{\rho}^{\,\ell}(1-\widehat{\rho})^{w-\ell}.$$

□

Proposition 6. ***Probability to wrongly guess a 0-coefficient***

Let j such that $r_j^{(i)}=0$. Then, the probability that $\widehat{r}_j^{(i)}=1$ is defined as

$$\tau_{0\to 1}=\sum_{\ell=\beta}^{w}\binom{w}{\ell}\widehat{\rho}^{\,\ell}(1-\widehat{\rho})^{w-\ell},$$

where

$$\widehat{\rho}=(1-\widetilde{\rho})\sum_{\ell=0}^{t-1}\binom{n_i-1}{\ell}\rho_z^{\ell}(1-\rho_z)^{n_i-1-\ell}+\widetilde{\rho}\sum_{\ell=t}^{n_i-1}\binom{n_i-1}{\ell}\rho_z^{\ell}(1-\rho_z)^{n_i-1-\ell},$$

with d_i, n_i and $t=\lfloor\frac{d_i-1}{2}\rfloor$ denoting the minimum distance, the length and the error correcting capability of the inner code, respectively, and

$$\begin{aligned}\widetilde{\rho}=\widetilde{\rho}_{w,w_r+1}&\left(1-\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)-(1-\rho_{w,w_r})\frac{w_e}{n}\right)\\&+(1-\widetilde{\rho}_{w,w_r+1})\left(\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)+(1-\rho_{w,w_r})\frac{w_e}{n}\right),\end{aligned}$$

$$\widetilde{\rho}_{w,w_r+1}=\sum_{\substack{\ell\in[1;\min\{w,w_r+1\}]\\ \ell\ odd}}\frac{\binom{w}{\ell}\binom{n-w}{w_r+1-\ell}}{\binom{n}{w_r+1}}.$$

Proof. The case $r_i^{(2)}=0$ is dealt analogously to that of a 1-coefficient. Firstly, according to Proposition 1, we can define:

$$\widetilde{\rho}_{w,w_r+1}=\sum_{\substack{\ell\in[1;\min\{w,w_r+1\}]\\ \ell\ \text{odd}}}\frac{\binom{w}{\ell}\binom{n-w}{w_r+1-\ell}}{\binom{n}{w_r+1}}.$$

Then, we have that $\widetilde{z}_j$ is equal to 1 with probability:

$$\widetilde{\rho} = \widetilde{\rho}_{w,w_r+1}\left(1-\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)-(1-\rho_{w,w_r})\frac{w_e}{n}\right) + (1-\widetilde{\rho}_{w,w_r+1})\left(\rho_{w,w_r}\left(1-\frac{w_e}{n}\right)+(1-\rho_{w,w_r})\frac{w_e}{n}\right),$$

Repeating the previous reasoning we obtain:

$$P(\widehat{e}_j = 1) = P(\perp_{\text{Inner}}(j) \mid z_j = 0)\cdot P(\widetilde{z}_j = 0) + P\big(\overline{\perp}_{\text{Inner}}(j) \mid z_j = 1\big)P(\widetilde{z}_j = 1). \tag{6.5}$$

Then, indicating $\widehat{\rho} = P(\widehat{e}_j = 1)$, by going over the same thought process again, we can express (6.5) as:

$$\widehat{\rho} = (1-\tilde{\rho})\sum_{\ell=t}^{n_i-1}\binom{n_i-1}{\ell}\rho_z^{\ell}(1-\rho_z)^{n_i-1-\ell} + \tilde{\rho}\sum_{\ell=0}^{t-1}\binom{n_i-1}{\ell}\rho_z^{\ell}(1-\rho_z)^{n_i-1-\ell},$$

and finally

$$\tau_{0\to 1} = \sum_{\ell=\beta}^{w}\binom{w}{\ell}\widehat{\rho}^{\,\ell}(1-\widehat{\rho})^{w-\ell}.$$

□

Using the above results, we are ready to derive the weight distribution for each polynomial $\Delta\mathbf{r}^{(i)}$.

Proposition 7. ***Weight distribution of*** $\Delta\mathbf{r}^{(i)}$

The probability that $\Delta\mathbf{r}^{(i)}$ *has weight* $w_r^{(i)}$ *is*

$$P\big(\mathrm{wt}(\Delta\mathbf{r}^{(i)}) = w_r^{(i)}\big) = \sum_{j=0}^{\min\{w_r, w_r^{(i)}\}} P(N_1 = j)P\big(N_0 = w_r^{(i)} - j\big),$$

where

$$P(N_0 = j) = \binom{n-w_r}{j}(\tau_{0\to 1})^j(1-\tau_{0\to 1})^{n-w_r-j},$$

$$P(N_1 = j) = \binom{w_r}{j}(\tau_{1\to 0})^j(1-\tau_{1\to 0})^{w_r-j}.$$

Proof. Whenever a one-coefficient of $\mathbf{r}^{(2)}$ is guessed correctly, the weight of $\Delta\mathbf{r}^{(2)}$ decreases by 1, while a wrong estimate does not change the weight of $\Delta\mathbf{r}^{(2)}$ in comparison with $\mathbf{r}^{(2)}$. Analogously, guessing a zero-coefficient correctly does not change the weight, while a wrong guess increases the weight by 1. Let N_0 denote the number of wrongly guessed 0-coefficients, and N_1 that of wrongly guessed 1-coefficients. Then, the weight of $\Delta\mathbf{r}^{(2)}$ is $N_0 + N_1$, and

$$P\big(\mathrm{wt}(\Delta\mathbf{r}^{(i)}) = w_r^{(i)}\big) = \sum_{j=0}^{\min\{w_r, w_r^{(i)}\}} P\big(N_1 = j, N_0 = w_r^{(i)} - j\big).$$

To conclude the proof, it is enough to assume that coefficients are guessed independently, so that both N_0 and N_1 are the sum of Bernoulli variables with respective parameters $\tau_{0\to1}$ and $\tau_{1\to0}$. □

To thoroughly validate the aforementioned propositions, consult Figure 6.3 and Figure 6.4. In these figures we juxtapose theoretical predictions with results obtained from numerical simulations. These simulations were performed using the HQC Round-I parameters recommended to achieve a 128-bit security level (as detailed in subsection 5.3.1).

As the figures clearly highlight, the proposed decoder seems to be very promising in reducing the DFR. We can select threshold values to keep the probabilities of incorrect guesses, represented by $\tau_{1\to0}$ and $\tau_{0\to1}$, very low. For example, an excellent choice is $\beta \in [30, 40]$. Also, we see that the reduction in the overall noise affecting the codeword can become very important. For instance, setting $\beta = 40$, we have a very large probability that $\Delta\mathbf{r}^{(i)}$ has null weight: with large probability, all coefficients have been correctly guessed. These results suggest that, considering our approach, the DFR of HQC can be strongly reduced. The obtained results are entirely general, and it is legitimate to assume that the same reasoning applies to the subsequent rounds of HQC in the NIST competition. In the next chapter, this will be shown, even when considering the HQC version with Reed-Solomon and Reed-Muller codes.

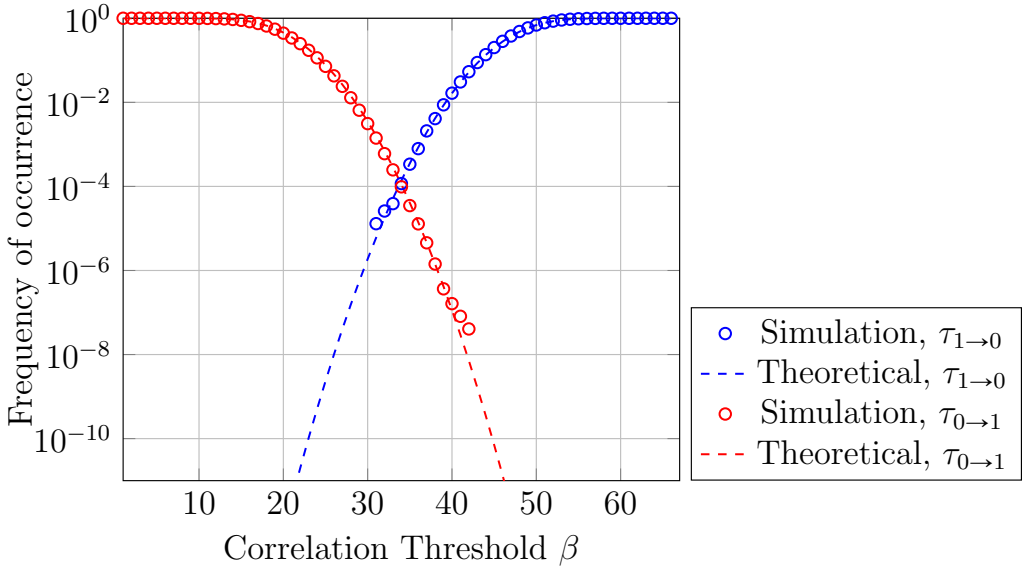

Figure 6.3: Values of $\tau_{1\to0}$ and $\tau_{0\to1}$, comparison between theoretical estimates and results of numerical simulations. For this experiment, we have considered the HQC parameters for the 128 bits of security (i.e., $n_1 = 796$, $n_2 = 31$, $w = 67$, $w_r = w_e = 77$. The empirical estimates have been obtained averaging over 10^3 trials.

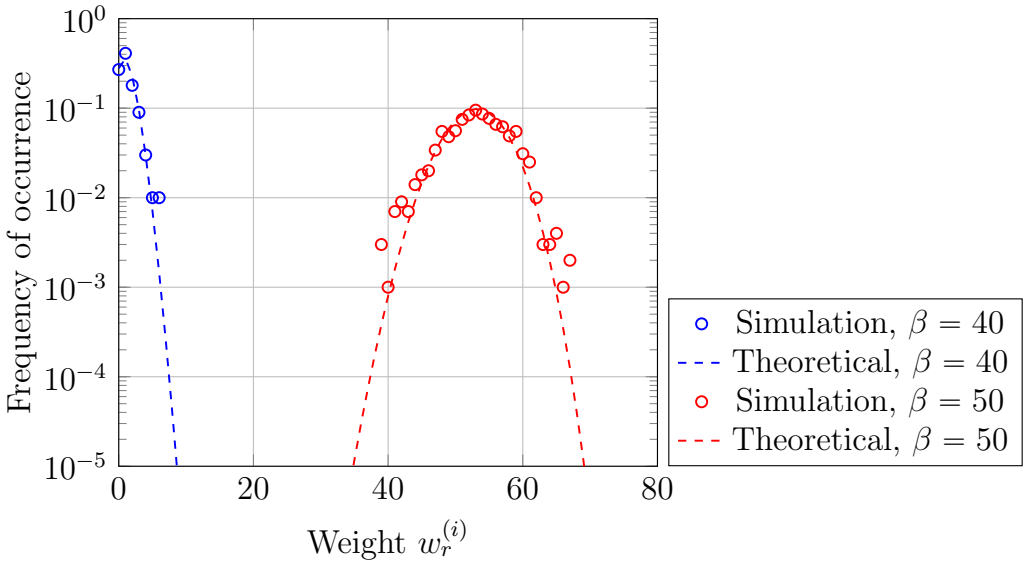

Figure 6.4: Weight distribution of $\Delta\mathbf{r}^{(i)}$, comparison between theoretical estimates and results of numerical simulations. For this experiment, we have considered the HQC parameters for the 128 bits of security (i.e., $n_1 = 796$, $n_2 = 31$, $w = 67$, $w_r = w_e = 77$. The empirical estimates have been obtained averaging over 10^2 trials.

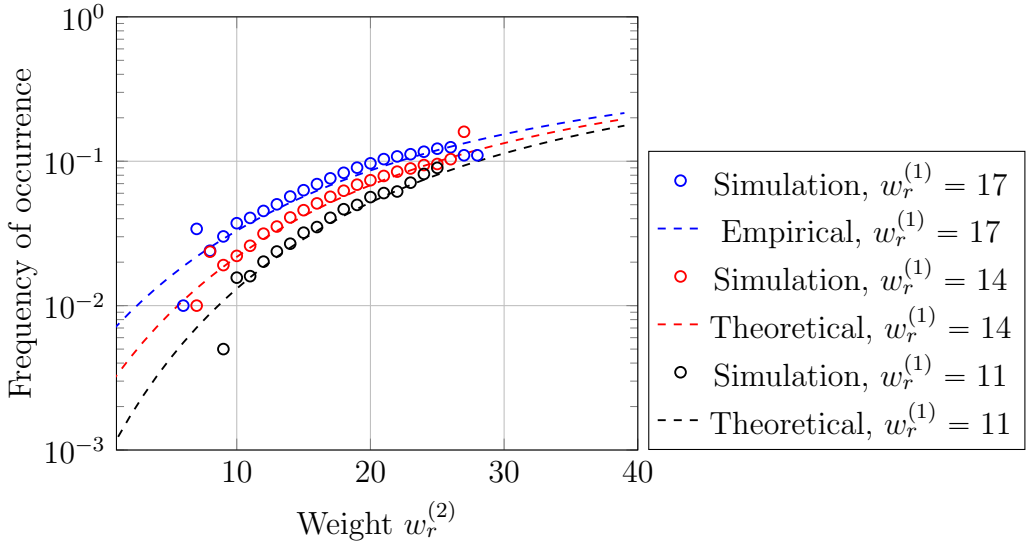

Figure 6.5: Probability to have a decoding error for a codeword of the repetition code. The considered parameters are $n_1 = 100$, $w = 17$, $w_r = w_e = 20$, $\beta = 12$, as a function of the weights of $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$.

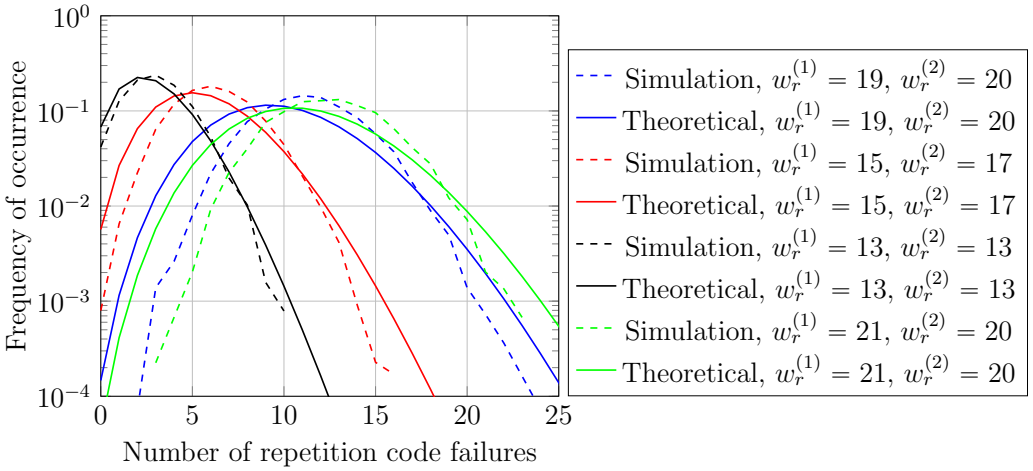

Figure 6.6: Probability distribution for the number of repetition code decoding errors. The considered parameters are $n_1 = 100$, $w = 17$, $w_r = w_e = 20$, $\beta = 12$, as a function of the weights of $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$.

An in-depth analysis of Figure 6.5 underscores a notable trend: the likelihood of encountering a decoding error for a codeword within the repetition code diminishes significantly as the weight of $\Delta\mathbf{r}^{(i)}$ decreases. Moreover, as the weight of $\Delta\mathbf{r}^{(i)}$ escalates, it inevitably leads to a proportionately heightened probability of experiencing a decoding error for a codeword associated with the repetition code. Turning our attention to Figure 6.6, this graph offers valuable insights into the probability distribution concerning the number of repetition code decoding errors. Notably, it illustrates that as the weight of $\Delta\mathbf{r}^{(i)}$ increases, the likelihood of encountering multiple repetition code failures becomes more pronounced. This observation highlights the sensitivity of the repetition code to variations in the weight of $\Delta\mathbf{r}^{(i)}$.

To study the DFR of our proposed decoder, we consider the following proposition. As we have seen, when the standard HQC decoder fails to decode $\Delta\mathbf{z}$ we have a decoding failure for our decoder. In Proposition 7, we explained how to determine the statistical distribution for the weights of $\Delta\mathbf{r}^{(1)}$ and $\Delta\mathbf{r}^{(2)}$. This approach should also help us establish the weight distribution of $\Delta\mathbf{z}$.

Proposition 8. ***DFR of the new decoder***

The DFR of the new decoder is

$$\epsilon = \sum_{w_r^{(1)}} \sum_{w_r^{(2)}} \epsilon\left(w_r^{(1)}, w_r^{(2)}\right) \cdot P\left(\mathrm{wt}(\Delta\mathbf{r}^{(1)}) = w_r^{(1)}\right) \cdot P\left(\mathrm{wt}(\Delta\mathbf{r}^{(2)}) = w_r^{(2)}\right).$$

Proof. Recalling Proposition 3, the DFR of the new decoder can be derived from the standard analysis of HQC considering $\Delta\mathbf{r}^{(i)}$ instead of $\mathbf{r}^{(i)}$. Hence:

$$\epsilon = \sum_{w_r^{(1)}} \sum_{w_r^{(2)}} \epsilon\left(w_r^{(1)}, w_r^{(2)}\right) \cdot P\left(\mathrm{wt}(\Delta\mathbf{r}^{(1)}) = w_r^{(1)}\right) \cdot P\left(\mathrm{wt}(\Delta\mathbf{r}^{(2)}) = w_r^{(2)}\right).$$

□

Chapter 7

Empirical results

In this chapter, we present the results of simulations conducted with the new decoder under various configurations. The simulations for the new decoder meticulously replicate the settings and parameters of HQC as observed during different rounds of the NIST competition. To elucidate, we scrutinize two distinct cases: the first entails the use of BCH and REP codes, mirroring the approach in subsection 5.3.1, while the second adopts Reed Solomon and Reed-Muller codes, echoing the methods in subsection 5.3.3. Moreover, we will provide simulations that consider the introduction of a new configuration. In this new setup, we employ a Reed-Solomon code as the outer code and replace the Reed-Muller code with a repetition code as the inner code. Additionally, we explore scenarios with soft Reed-Muller code decoding and an iterative approach with multiple noise-filtering stages.

7.1 BCH+REP

In Figure 7.1, we show the DFR values for the new decoder as we vary the threshold T while also varying the REP length n_2. Notably, the figure shows that the optimal threshold value for this HQC setup is T = 38, given that other parameters being equal it allows to obtain lower DFR.

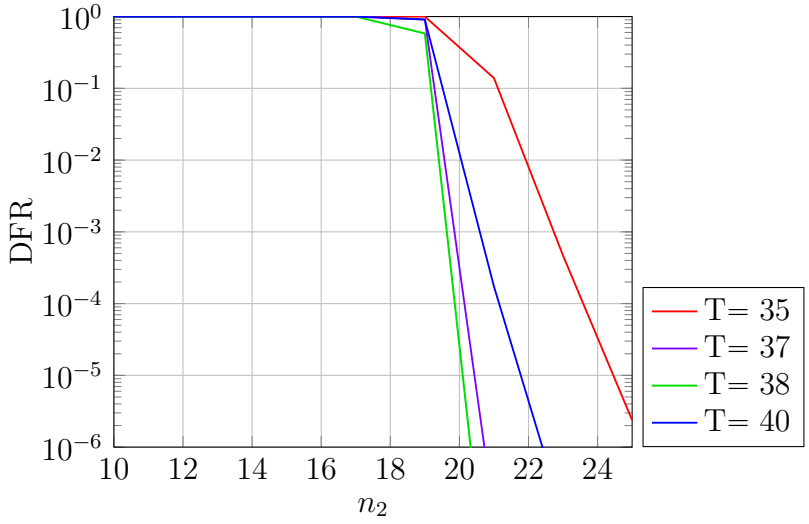

Figure 7.1: DFR for the proposed decoder with parameters $n_1 = 766$ (BCH length), $w = 67$, $w_r = w_e = 77$ and $t_{BCH} = 57$.

Furthermore, it is of great significance to conduct a detailed comparison of the DFR values acquired in this particular configuration with those achieved using the original HQC decoder under identical parameters and conditions. This comparative analysis serves to unequivocally highlight the distinct advantages our decoder offers in contrast to the original decoder. The graphical representation of this comparison can be seen in Figure 7.2

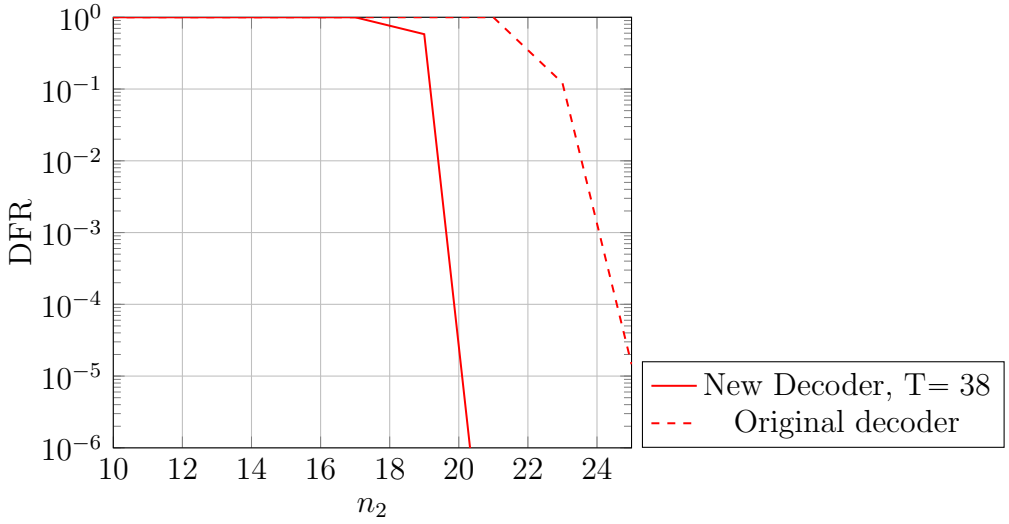

Figure 7.2: DFR for the proposed decoder with parameters $n_1 = 766$ (BCH length), $w = 67$, $w_r = w_e = 77$, $t_{BCH} = 57$ and T= 38 and for the original HQC decoder with same parameters (HQC round 1 with SL=128)

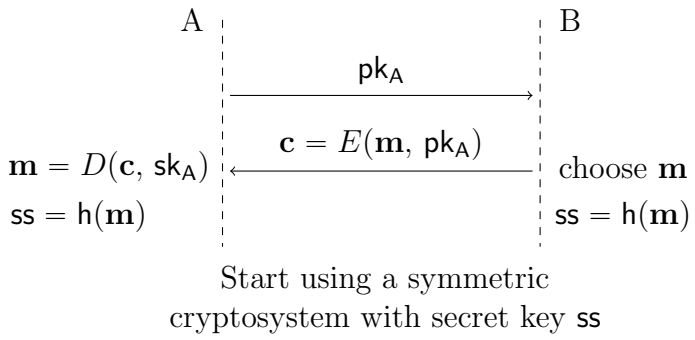

Figure 7.3: Classic hybrid encryption scheme. Specifically, a public key cryptosystem is used solely for key exchange, after which the actual encryption (bulk encryption) is performed through symmetric encryption.

In a practical context, HQC in its KEM version (section 5.2) can be used as a public key cryptosystem in hybrid encryption (Figure 7.3). In cryptographic protocols, a Key Encapsulation Mechanism serves the purpose of safeguarding symmetric key material for transmission through the utilization of public-key algorithms. This approach is commonly employed within hybrid cryptosystems. In practical applications, public key systems are not particularly efficient when it comes to transmitting lengthy messages. Instead, they are typically employed for the exchange of relatively short symmetric keys. The symmetric key is subsequently utilized to encrypt the longer message. The conventional method for transmitting a symmetric key within public key systems involves the initial generation of a random symmetric key, which is then encrypted using the chosen public key algorithm. The recipient can then decrypt the public key message to retrieve the symmetric key. Specifically, let's consider two parties, referred to as A and B. Initially, A sends his public key to B (we will overlook the secure distribution of public keys in this context). Once B receives A's public key, he chooses a message, denoted as $\mathbf{m}$ which he encrypts using A's public key. The message, encrypted with A's public key, can only be decrypted by A, as he is the only one in possession of the corresponding secret key. At this point, A decrypts the message and retrieves the original $\mathbf{m}$. From this moment on, A and B agree to treat $h(\mathbf{m})$ as their shared secret key,

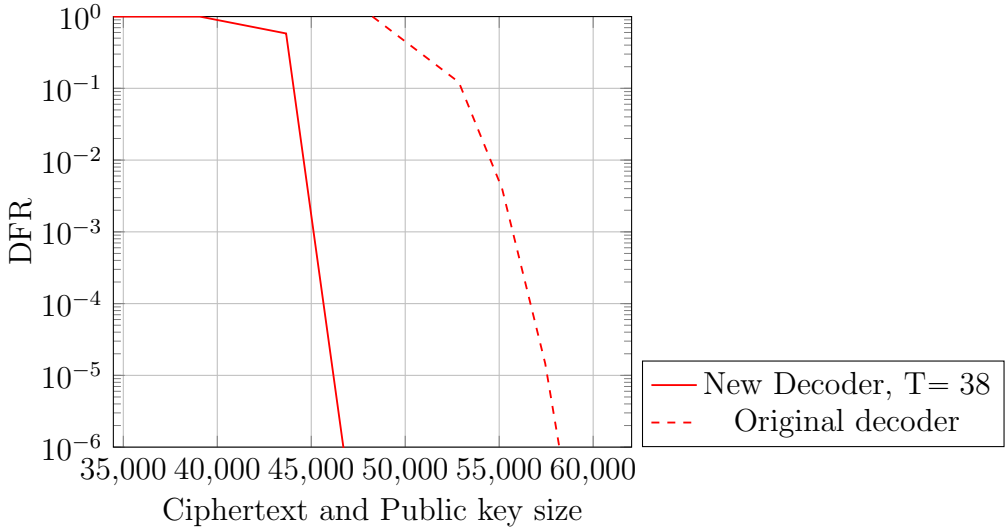

Figure 7.4: Comparison of the DFR of the new HQC decoder with T=38 and the original decoder as the repetition code length (n_2) varies. The parameters are the same as in Figure 7.2, with the only difference being the x-axis. Specifically, the x-axis represents the size in bits of the ciphertext plus the size in bits of the public key.

where $h(\cdot)$ represents a generic hash function. Once the shared secret key is indeed known to both A and B, they can begin encrypting using a symmetric cryptosystem, such as AES (Advanced Encryption Standard) [32].

In the broadest context, this discussion can certainly be customized to the particular case of HQC. Consequently, it becomes evident that in this specific scenario, the data size transferred over the public channel corresponds to the combination of the public key size and the ciphertext size. In light of this, within several of the graphs provided below, when considering this precise context, the x-axis is employed to depict the dimensions of both the public key and the ciphertext. Both of these dimensions are evidently linked to n which is assumed to be the length of the public code employed. As a consequence, it becomes clear that decreasing the value of n while upholding a consistent DFR naturally leads to a reduction in the size of the data conveyed via the public channel. In this context, Figure 7.4 undeniably emphasizes the benefits we achieve in terms of reducing both the dimensions of the ciphertext and the public key when contrasted with the original HQC decoder.

In Table 5.10, we provide a detailed specification of the sizes for the ciphertext and the public key. Clearly, in this scenario, with the assumption that $n_1 = 766$, we can deduce that $n \approx 766 * n_2$. Therefore, the combined size of the ciphertext and public key amounts to approximately $3n \approx 2298 \cdot n_2$ (neglecting constant terms). To put it differently, Figure 7.4 can be regarded as nothing more than a duplicate of figure 7.2, with a suitable adjustment of the x-axis scale. Certainly, by setting the DFR value to a threshold of interest, it's possible to significantly reduce the value of n and, consequently, the size of the data transmitted over the public channel. For example, let's imagine a hypothetical situation where we assume a DFR of 10^{-3}. When transitioning from the original decoder to the new decoder, we can reduce the number of bits from approximately 56,000 to 45,000. This means that there is a reduction of approximately 1375 bytes concerning the data that travels on a public channel, explicitly, in percentage terms, the reduction is about 20%. The same line of reasoning can be extended to the HQC parameters aimed at achieving security levels of 192 and 256 (refer to Table 5.2). Undoubtedly, in these specific scenarios, it remains feasible to establish a direct correlation between the decrease in the variable n_2 and the consequent reduction in the size of the data transmitted over the public channel.

7.2 RS+RM

Now, we aim to revisit the insights presented in the preceding section, focusing on scenarios involving Reed-Solomon and Reed-Muller codes. Formally, all the considerations discussed so far remain applicable, as they are entirely code-agnostic. What remains to be addressed is quantifying the improvement provided by the proposed decoder in comparison to the original decoder when Reed-Solomon and Reed-Muller codes were employed, as in the third and fourth rounds of the NIST competition.

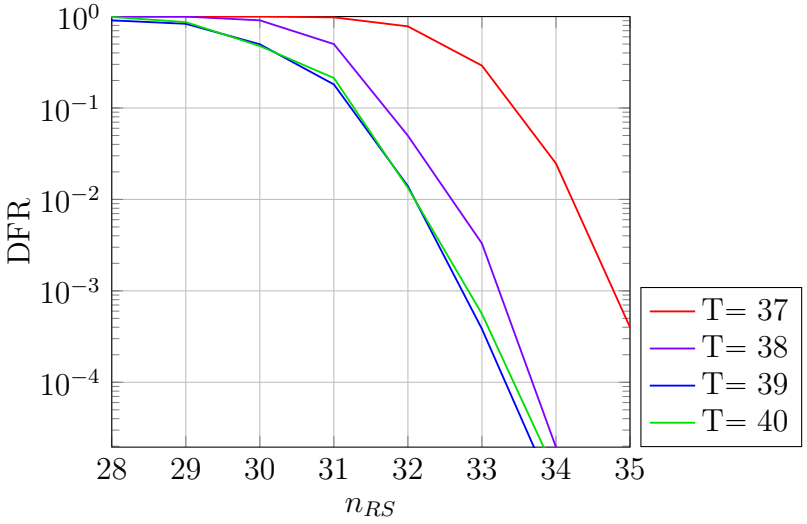

Figure 7.5: DFR for the proposed decoder with parameters $n_{RM} = 128$ (original length RM code), $mult = 3$ (multiplication factor RM code), $w = 66$, $w_r = w_e = 75$ (HQC parameters for round 3 and 4 achieving SL=128)

In Figure 7.5, we present a visual representation of the DFR values for the newly developed decoder. Our primary objective is to scrutinize the fluctuations in these DFR values as we fine-tune two critical parameters: the threshold value T and the Reed-Solomon length n_{RS}. What makes this graphical representation particularly noteworthy is its capacity to vividly showcase that, when the parameters are held constant, the most favorable choice for the threshold value is T = 39. This specific threshold value consistently results in a lower DFR, thereby underscoring its pivotal role in enhancing overall performance. This finding offers valuable insights into the optimization of these parameters and their impact on the efficiency of the decoder.

Furthermore, conducting a comprehensive comparison between the DFR values achieved in this particular configuration and those obtained when employing the original HQC decoder under the same parameters and conditions is of significant importance. This comparative evaluation plays a crucial role in clearly illustrating the distinct advantages that our decoder offers in comparison to the original decoder. The visual representation of this comparative analysis can be found in Figure 7.6.

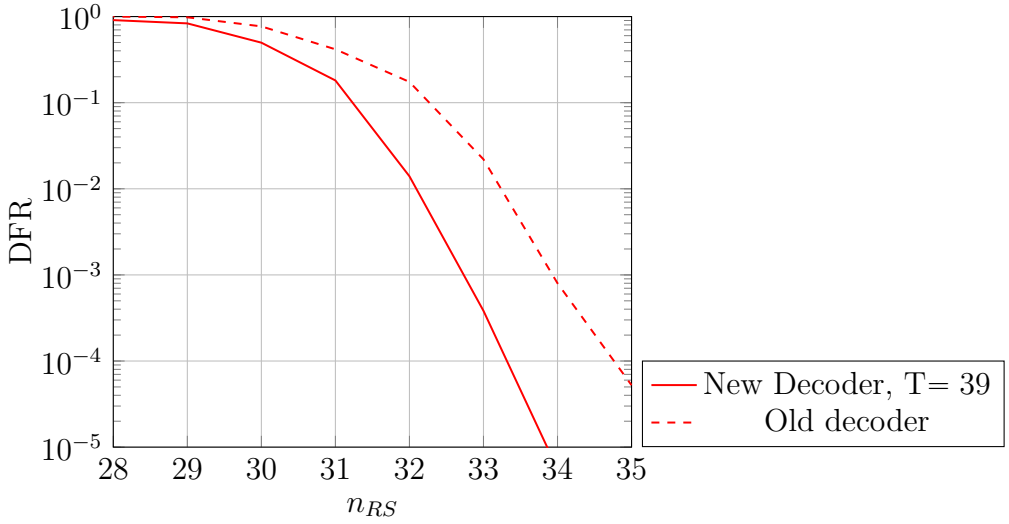

Figure 7.6: DFR for the proposed decoder with parameters $n_{RM} = 128$ (original length RM code), $mult = 3$ (multiplication factor RM code), $w = 66$, $w_r = w_e = 75$ and T= 39 and for the original HQC decoder with same parameters (HQC round 3 and 4 with SL=128)

Figure 7.7 shows the advantages we gain in terms of reducing both ciphertext and public key sizes when compared to the original HQC decoder. In this scenario, assuming $n_{RM} = 128$ and $mult = 3$, we can deduce that n is approximately $384 \cdot n_{RS}$. Consequently, the combined size of the ciphertext and public key approximates to $3n \approx 1152 \cdot n_{RS}$. As described in the previous section, by selecting a DFR threshold of interest, it becomes possible to reduce the value of n, and consequently, the volume of data transmitted over the public channel. For instance, consider a hypothetical scenario where we assume a DFR of 10^{-3}. When switching from the original decoder to the new decoder, we can reduce the number of bits from around 39,000 to 37,500. This translates to a reduction of approximately 188 bytes in the data transmitted over the public channel, equivalent to a roughly 4% decrease. Furthermore, this line of reasoning extends itself seamlessly when we contemplate HQC parameters aimed at achieving security levels of 192 and 256. In these scenarios as well, we can discern a direct correlation between the decrease in n_{RS} and the concurrent reduction in the size of the public data.

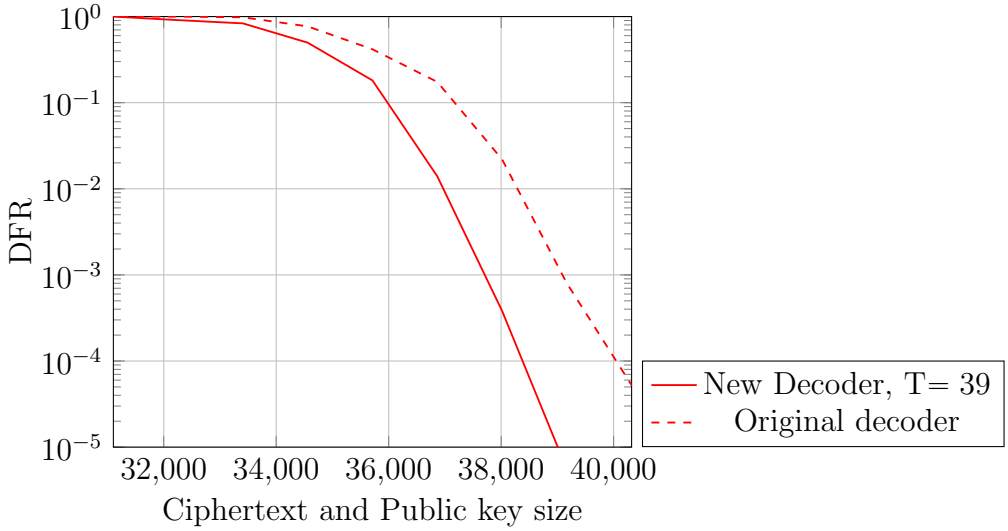

Figure 7.7: Comparison of the DFR of the new HQC decoder with T=39 and the original decoder as the RS code length (n_{RS}) varies. The parameters are the same as in Figure 7.6, with the only difference being the x-axis. Specifically, the x-axis represents the size in bits of the ciphertext plus the size in bits of the public key.

7.3 RS+REP

In addition to the previously discussed configurations, we have explored a scenario where we replaced the Reed-Muller code with a repetition code. This involved utilizing a concatenated code as the public code, with the outer code being the traditional Reed-Solomon code, similar to the third and fourth rounds of the NIST competition, while the inner Reed-Muller code was replaced with a repetition code. Each RS code symbol is initially transformed into an 8-bit binary string, and then these bits are duplicated according to the repetition code. To align with the Reed-Solomon and Reed-Muller setup, we chose a repetition code with a length of $n_{REP} = 48$. This choice allows us to replicate each of the 8 bits associated with a single Reed-Solomon code symbol 48 times, resulting in a codeword length that aligns with what would be achieved if the Reed-Muller code were employed as the inner code instead of a repetition code. Moreover, it is important to emphasize that the repetition code is decoded using the standard majority decoding method.

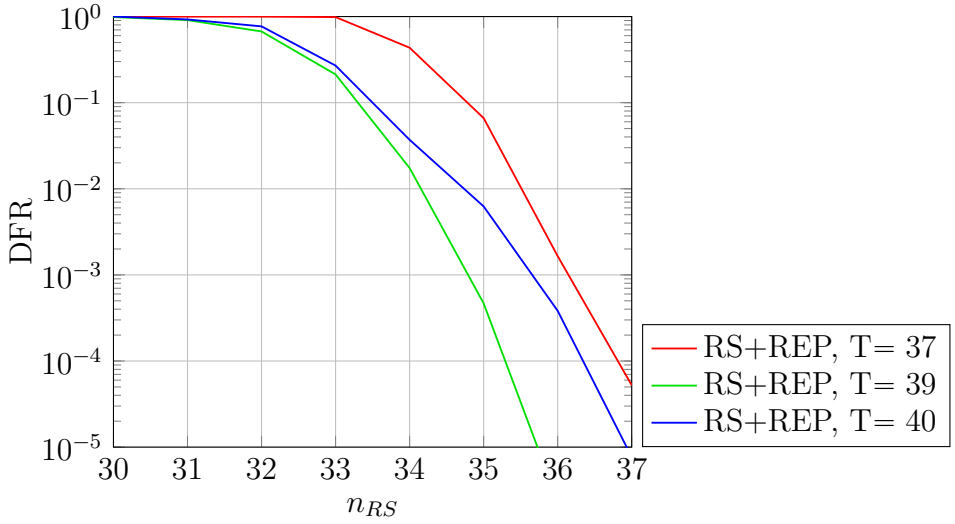

Figure 7.8: DFR for the proposed decoder in case of using Reed-Solomon as outer code and a repetition code as inner code. This simulation is conducted with the following parameters: $n_{rep} = 48$, $w = 66$ and $w_r = w_e = 75$

In Figure 7.8, we present the DFR values for the new decoder. Specifically, our objective is to observe how these values vary as we make adjustments to the threshold value. It's important to note that the graph is plotted while varying the length of the Reed-Solomon code and adjusting the threshold. The primary goal is to determine the optimal threshold value. Notably, in this particular scenario, the optimal threshold choice is T=39. It is evident that we have a keen interest in assessing the performance achieved in this particular scenario as compared to that attainable with the standard HQC approach. The conventional HQC approach, inclusive of BCH and repetition codes, alongside Reed-Solomon and Reed-Muller codes, serves as a comprehensive reference point. In Figure 7.9, we have depicted this comparison.

From Figure 7.9 , it is clear that the best performance is achieved in the HQC version with Reed-Solomon and Reed-Muller codes using our proposed decoder with the optimal threshold of T=39, as shown in Figure 7.5. The performance when using a repetition code in place of the RM code is lower but still better than what can be achieved using BCH and repetition codes as in the first version of HQC.

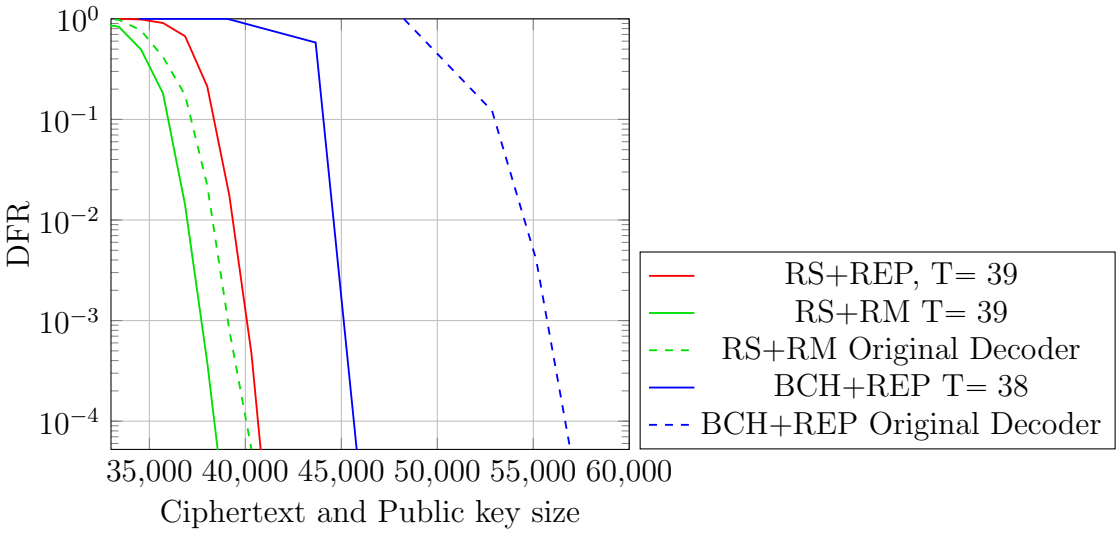

Figure 7.9: In this figure, we're examining the DFR across three distinct scenarios. The first scenario involves the use of BCH and repetition codes, with parameters $n_1 = 766$ (BCH length), $w = 67$, $w_r = w_e = 77$, $t_{BCH} = 57$, and T=38. This configuration corresponds to HQC round 1 with a security level of 128 (SL=128). The second scenario features Reed-Solomon and Reed-Muller codes, with parameters $n_{RM} = 128$ (original RM code length), $mult = 3$ (RM code multiplication factor), $w = 66$, $w_r = w_e = 75$, and T= 39. These settings are representative of HQC rounds 3 and 4 with a security level of 128 (SL=128). The third and final scenario employs Reed-Solomon and repetition codes, with parameters $n_{rep} = 48$, $w = 66$, $w_r = w_e = 75$, and T= 39.

Furthermore, Figure 7.9 highlights the advantage gained by opting for Reed-Solomon and Reed-Muller codes instead of BCH and repetition codes. In particular, it is worth reiterating that the best performance is achieved by considering HQC in its latest version presented in the NIST competition, especially with the inclusion of our proposed decoder.

7.4 More than one filtering stage

Considering the computational cost associated with the new decoder (described in detail in section 6.2), which exhibits a relatively modest increase, specifically, it scales linearly with n, it is clear that introducing an iterative decoder that encompasses multiple noise filtering steps is feasible possible. As depicted in Figure 6.2, our proposed decoder can be readily seen as the conventional HQC decoder with a noise filtering stage applied as a pre-processing step. Naturally, there is no restriction on the number of iterations of this noise filtering step, but it's evident that this will inevitably lead to an increment in computational cost. In the following discussion, we aim to emphasize this specific scenario, demonstrating that such an iterative approach yields performance improvements that, while present, are not substantially significant. Consequently, as will be illustrated in this section, introducing iterative decoding may not be warranted. In other words, the enhancement in performance may not justify the corresponding increase in complexity to a significant extent.

In Figure 7.10, we present a simulation that mirrors the parameters utilized in Figure 7.5, while introducing the concept of employing multiple noise filtering stages. The aim is to delve into the impact of successive noise filtering on the performance, thereby assessing the effectiveness of this iterative approach within the same parameter framework as previously analyzed.

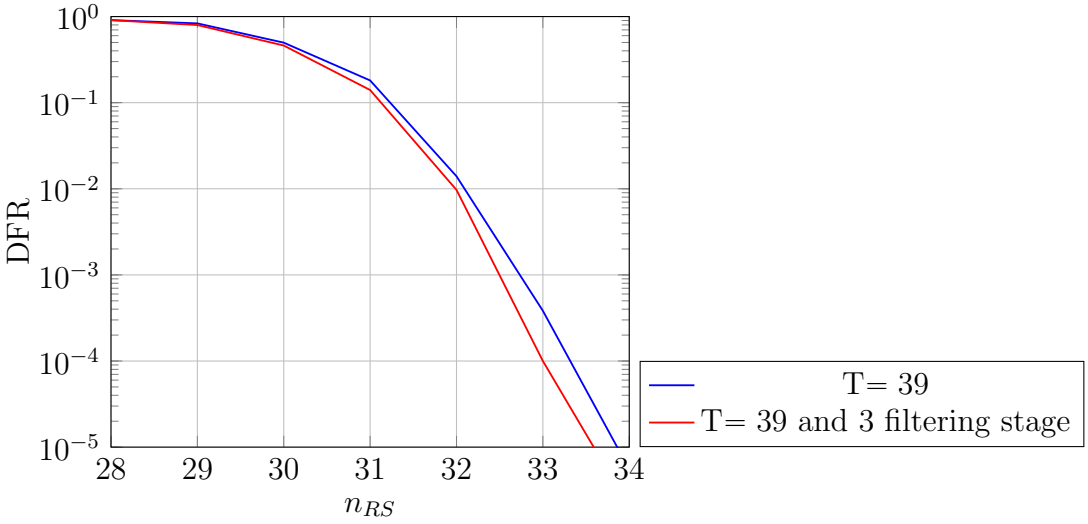

Figure 7.10: DFR comparison for the proposed decoder with parameters $n_{RM} = 128$, $mult = 3$, $w = 66$, $w_r = w_e = 75$ and $T = 39$ when employing one versus three filtering stages.

As we anticipated, the data presented in Figure 7.10 underscores that the benefits gained from employing multiple filtering stages are quite marginal. Upon closer examination, it becomes apparent that the initial filtering step significantly reduces the error's weight and establishes a robust correlation with the initial error pattern. This intrinsic correlation between the residual error and the initial error pattern makes any subsequent filtering steps less effective in enhancing overall performance. It's analogous to the idea that what the first filtering step is unable to correct, subsequent steps also struggle to rectify. This observation highlights the fact that the overall improvement achieved through multiple iterations of the initial filtering is remarkably minimal. This marginal gain is insufficient to practically justify the added complexity and computational overhead that comes with implementing multiple filtering stages. In essence, it signifies that the benefits of this approach do not reach a level that would make it a viable and efficient strategy in practice.

Chapter 8

Applications of the New Decoder

In our previous discussion, we derived the theoretical DFR for the new decoder (refer to section 6.3). Furthermore, we conducted extensive simulations in both C and Python to evaluate the relationship between the theoretical and the simulated DFR curve. Figure 8.1 illustrates that the theoretical curve serves as a highly accurate approximation of the simulated one. However, it is evident that the theoretical curve exhibits a slightly more optimistic outlook compared to the simulated curve. Consequently, we have not been able to identify a mathematical upper bound for the DFR, making it currently impossible to formally demonstrate the IND-CCA property. These same observations apply to various versions of HQC with different parameter sets.

While the theoretical curve remains a valuable metric, enriching our understanding of the decoder's performance, a critical and rather technical necessity arises in the quest to establish an upper bound for the DFR, particularly for proving the IND-CCA (and by extension IND-CCA2) property. Indeed, to formally prove that IND-CCA holds, one needs a provable upper bound on the success probability: without such a bound, the proof cannot be produced and IN-CCA cannot be claimed.

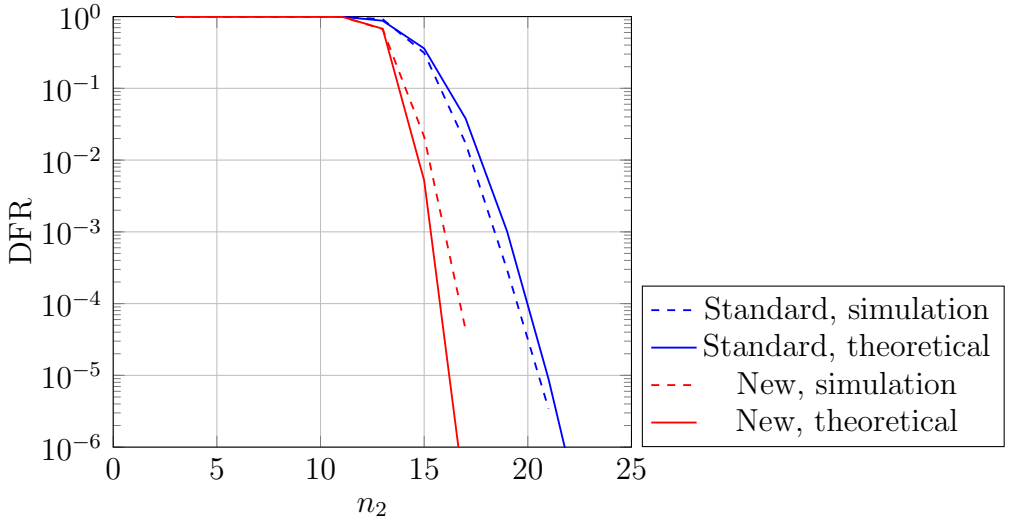

Figure 8.1: DFR for HQC with parameters $n_1 = 100$, $w = 17$, $w_r = w_e = 20$, $\beta = 12$.

Nonetheless, it is of paramount importance to underscore that specific applications exclusively require adherence to the IND-CPA property. The recognition of these particular contexts serves to highlight the practical significance of our findings. This recognition implies that, although the demonstration of the IND-CCA property may pose certain challenges, the fulfillment of the IND-CPA property might be adequate for targeted application scenarios. This acknowledgment not only solidifies the foundation of our research but also lays the groundwork for a meticulous and comprehensive exploration of how our results support practical implications in real-world situations.

8.1 Use of Ephemeral Keys

Although the formal demonstration of an upper bound for the DFR remains pending, the outcomes derived from this thesis underscore the practical significance of our new decoder. Beyond being a mere mathematical exercise, the implications of this decoder are notable, particularly in scenarios where the IND-CPA property proves sufficient such as those involving the use of ephemeral keys. The utility of our decoder extends beyond theoretical con-

siderations, pointing towards its practical relevance in real-world applications. An ephemeral key is defined as one generated for each execution of a key-establishment process, meeting additional requirements of the key type, such as uniqueness for each message or session [33].

To illustrate this, let us consider a practical situation where an asymmetric cryptosystem facilitates the exchange of a session key, subsequently employed for symmetric encryption. Imagine communicators A and B attempting to agree on a key using an asymmetric cryptosystem. In this case, A, for instance, shares his public key with B. Then, B encrypts a secret key of his choice with A's public key and transmits the ciphertext to A. B is aware that only A can decipher the hidden message, i.e. the key, since only A possesses the secret key corresponding to the public key used by B for encryption.

In this scenario, it can be proved that a decryption failure discloses information about the key used during decryption. In other words, an attacker observing when decryption produces failures might gain significant insights into the key used for decryption, i.e., A's secret key. Decryption failures, in essence, act as inadvertent indicators that reveal details about the cryptographic operations at play. An adversary, monitoring these failures, could leverage the patterns and anomalies associated with unsuccessful decryptions to infer critical aspects of the cryptographic system. In this context, the vulnerability lies in the fact that the decryption failure itself can inadvertently disclose information about the key, allowing an attacker to piece together essential components of A's secret key. Ensuring a sufficiently low DFR is paramount, and our ultimate goal is to establish an upper bound, denoted as $DFR_{u.b.}$, satisfying the condition $DFR < DFR_{u.b.} \leqslant 2^{-\lambda}$, with λ being the chosen security parameter. This emphasis on determining the upper bound is particularly critical in scenarios susceptible to a Chosen Ciphertext Attack (CCA). In such instances, an adversary can inundate A with ciphertexts, strategically leveraging decryption failures to extract substantial information about A's secret key. Consequently,

the quest for the upper bound for the DFR plays a vital role in achieving Indistinguishability under Chosen Ciphertext Attack (IND-CCA) and its more robust version, IND-CCA2.

Given the preceding discussion, the adoption of ephemeral keys emerges as a crucial practice in certain applications. This approach entails user A decrypting messages with a secret key subject to systematic changes, primarily involving continuous key rotation. The implications of this key rotation, particularly in the context of decryption failures, are noteworthy. In the event of such failures, an attacker's access would be limited to information about a key that is no longer actively in use, rendering the obtained data essentially obsolete. This security measure aligns seamlessly with the concept of Perfect Forward Secrecy (PFS). Perfect Forward Secrecy (PFS) guarantees that even if an adversary captures and scrutinizes past ciphertexts, the acquired information corresponds to a key that has been replaced. The transient nature of ephemeral keys serves to increase security by restricting the exposure of sensitive information. Essentially, the dynamic changes in keys under PFS not only protect ongoing communications but also support the resilience of the cryptographic system against potential threats over time. This proactive key management approach plays a significant role in enhancing overall security robustness, making PFS a valuable feature in scenarios where continuous key renewal is of paramount importance.

Within this specific context, the immediate need doesn't necessitate establishing an upper limit for the DFR. Instead, the central focus is on ensuring that the DFR remains at a sufficiently low level. This strategic approach is designed to prevent any undue hindrance in the key exchange process, thereby safeguarding the efficiency and timely execution of cryptographic operations. The priority here is to strike a balance that optimally maintains the security protocols without introducing unnecessary delays in the cryptographic procedures.

Here, without assuming exhaustive coverage, we outline several scenarios where the application of ephemeral keys is envisioned. Consequently, in these contexts, the decoder proposed in this thesis can undoubtedly be considered an efficient solution.

8.1.1 TLS (Transport Layer Security)

Transport Layer Security (TLS), serves as a cryptographic protocol ensuring secure communication across computer networks. Throughout the Internet's existence, the persistent presence of security threats has prompted continual evolution in security protocols. Notably, Secure Sockets Layer (SSL), introduced by Netscape in 1995, played a pivotal role in this evolution. However, SSL's susceptibility to significant security vulnerabilities led to its restriction by the U.S. government in 2014 [34]. Consequently, the mandate shifted towards the adoption of the next-generation security protocol, Transport Layer Security. While TLS effectively mitigated the vulnerabilities present in SSL, the preceding versions of TLS, specifically those before version 1.3, encountered challenges related to performance issues linked to the extended handshake.

Under TLS 1.2, the handshake protocol required two request-response exchange to authenticate a client to the server. This version found widespread use on the web, securing all HTTPS websites and enabling secure communication for various protocols such as SMTPS for emails and FTPS for secure file transfers. The advent of TLS 1.3 allowed for the reduction of the key creation handshake to a single request and response round trip. This simplified procedure enables both the client and server to share the same encryption key, generated independently and simultaneously through the Ephemeral Diffie-Hellman (EDH) key exchange technique, among other possible methods. The term "ephemeral" underscores the temporary and dynamic nature of these keys, contributing to the robustness of the cryptographic process in TLS 1.3.

As previously mentioned, TLS 1.3 employs a method of encrypting data

that enables both the client and the server to encrypt data using a shared private key. Notably, this private key is never transmitted over the internet. The mechanism operates by having both the server and the client generate their unique secret key values, respectively denoted as x and y. Each party possesses its distinct secret key value. These individual secret key values, along with a public key known to both the client and server, are utilized to create the ephemeral symmetric key. In accordance with the classical Diffie-Hellman protocol, the public key is composed of a very large prime number p and a primitive root g. In the intricate TLS 1.3 handshake protocol, our focus in this section is specifically on the generation of the symmetric key. While the complete TLS handshake process is complex, we aim to provide insights into the crucial aspect of symmetric key generation without presuming to cover all details comprehensively.

In the TLS 1.3 handshake protocol, the server initiates the process by sharing its public key certificate and the signed key material with the client. This key material, denoted as $g^x \bmod p$ calculated using Diffie-Hellman, is signed to prevent potential Man-in-the-Middle (MITM) attacks. If the client also possesses a certificate, it replays by sending its certificate to the server along with the signed key material computed as $g^y \bmod p$. Following this, both parties calculate $g^{(x \cdot y)} \bmod p$ as the session key and begin encrypting data within that session using a shared key in a symmetric cipher. This key is temporary, changing with each session and initiating a fresh key exchange for each session. This process ensures Perfect Forward Secrecy (PFS), meaning that if the secret x (or y) is compromised, the impact is limited to the recovery of the symmetric key exchanged in that specific session, without affecting other past or future sessions. A similar process occurs in the TLS tunnel version, where the client, without a certificate, generates a key randomly. This key is then sent to the server, encrypted with the server's public key. Then, the server decrypt using his secret key in order to recover the symmetric session

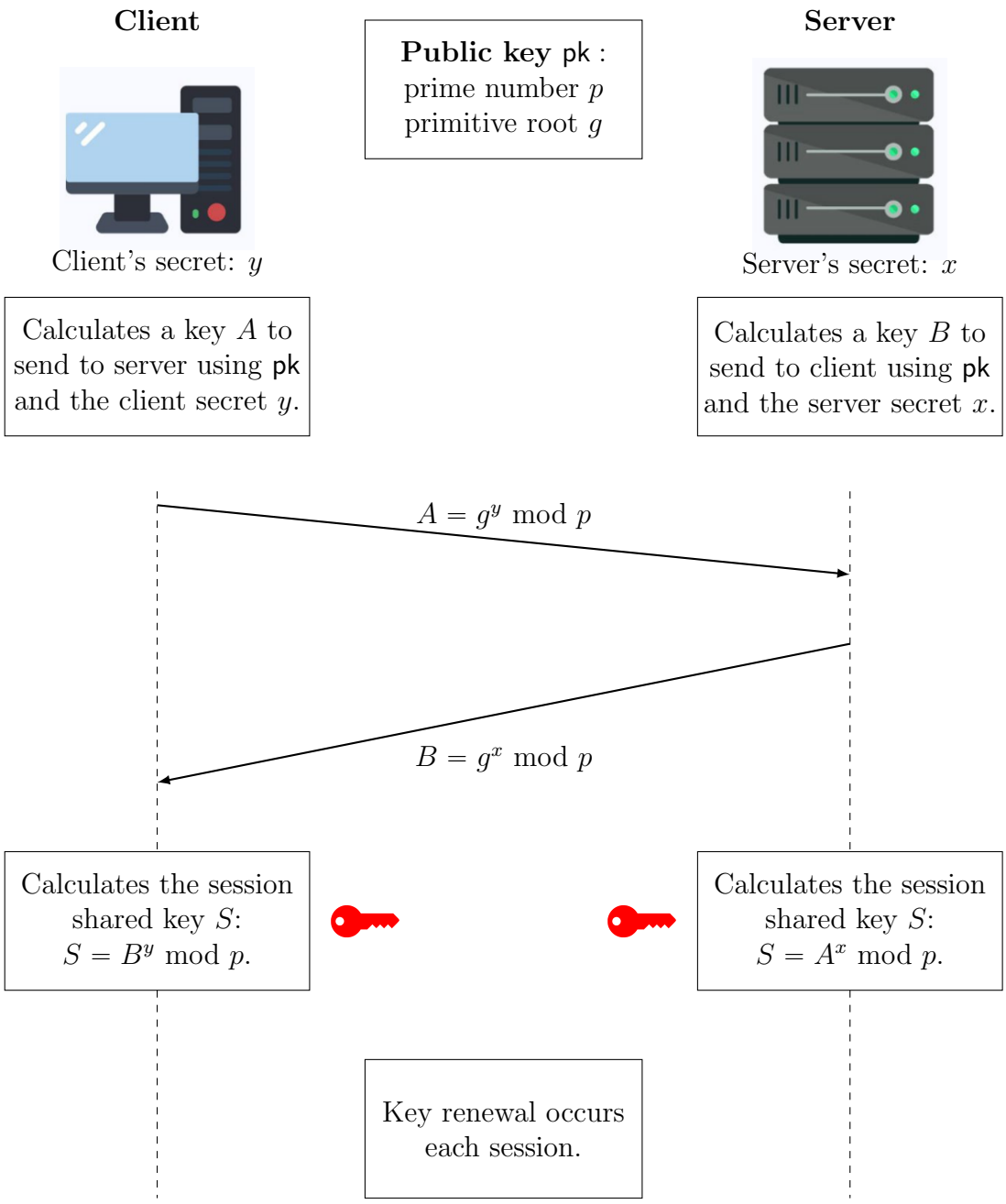

Figure 8.2: Key Exchange protocol (part of the Handshake protocol) using Ephemeral Diffie Hellman (EDH) in TLS 1.3

key. Even in this case, the relevant key undergoes modification with each session, upholding the principles of PFS. To visually comprehend what has been discussed so far, refer to Figure 8.2.

8.1.2 VPN (Virtual Private Network)

Virtual Private Networks (VPNs) are crucial for ensuring secure communication over networks, establishing a private and encrypted connection between users and remote servers. A VPN is a virtual network built on top of existing physical networks that provides a secure communications mechanism for data and control information transmitted between computers or networks [35]. In this subsection, we aim to delve deeper into the utilization of ephemeral keys within the framework of VPNs, expanding on the principles previously discussed in the context of Transport Layer Security (TLS) (refer to subsection 8.1.1). Our primary focus is to highlight their dynamic nature and their contributions to the overall security framework.

In a VPN, ephemeral keys enhance the security of data transmission by introducing temporary cryptographic keys with a limited lifespan. By employing keys that are short-lived and unique to each session, the VPN system ensures that the compromise of a single key does not jeopardize the security of past or future communications. In other words, the use of ephemeral keys in VPNs contributes to achieving Perfect Forward Secrecy (PFS).

VPN tunnel encryption is a method designed to safeguard data transmitted between two endpoints within a virtual private network, shielding it from unauthorized interception, modification, or theft. While VPN tunnel encryption is a robust security measure, it is not infallible. In the unfortunate event that an adversary successfully compromises the encryption key, the consequences are far-reaching, allowing him unauthorized access not only to the ongoing data exchange within the virtual private network but also to both historical and forthcoming data. This scenario underscores the imperative role assumed by Perfect Forward Secrecy. Specifically, PFS can be seen as a feature that enhances the security of VPN tunnel encryption by ensuring that each session possesses a unique and ephemeral key that cannot be deduced from prior or subsequent keys.

Various protocols for Virtual Private Networks (VPNs), such as OpenVPN and IKEv2/IPsec, incorporate advanced mechanisms for exchanging ephemeral keys within their frameworks. These mechanisms play a crucial role as foundational elements when initiating a VPN session, making a substantial contribution to the creation of resilient and secure communication channels. As the VPN session initiates, the ephemeral keys come into play by serving as the foundation for data encryption. These keys, dynamically generated for the specific session, are utilized to encrypt the data flow traversing the VPN tunnel. In the tunnel mode, the entire IP packet is encrypted and authenticated. It is then encapsulated in a new IP packet with a new IP header [36]. The tunnel mode is used to create virtual private networks for communication from one network to another (VPN), from a host to a network (remote access), and between two hosts (e.g., private chat). This encryption process adds an additional layer of protection to the transmitted information, ensuring that even if intercepted, the data remains unintelligible to unauthorized entities.

Chapter 9

Conclusion and future work

In conclusion, the central aim of this thesis has been to introduce an alternative decoding algorithm for HQC, driven by the general objective of enhancing its cryptographic performance. Through the course of this research, we have made significant progress in developing a decoder that demonstrates notable advancements over the capabilities of the original algorithm. This progress is not only supported by rigorous mathematical analysis but also confirmed through extensive simulations, emphasizing the strength and reliability of the proposed solution.

The key advantage of the newly proposed decoder is its ability to maintain a low Decoding Failure Rate (DFR) while enabling the use of shorter code lengths. In other words, the new decoder allows us to achieve the same DFR performance as the original decoder, demonstrating that it is just as reliable in error correction, while also enabling the use of shorter codes. The advantage of employing shorter codes, however, extends well beyond the confines of error correction. It provides a tangible and practical benefit by leading to a substantial reduction in the dimensions of both the public key and the ciphertext. This reduction in code length serves to enhance the overall efficiency and economy of cryptographic systems without compromising the security of the data they protect. Consequently, the capacity to reduce the size of cryptographic

keys and ciphertext while sustaining the same DFR performance represents a significant stride in the field of post-quantum cryptography. These findings hold substantial implications for the practical application of HQC in real-world cryptographic scenarios, where maintaining a delicate balance between efficiency and security is of paramount importance. Delving deeper into our proposal, the newly suggested decoder for HQC stands out as a notably more efficient solution, particularly in contexts where ephemeral keys come into play. This is especially relevant in security protocols like TLS or VPNs, where the utilization of ephemeral keys is a common practice. On top of that, it is worth emphasizing that our solution extends its efficiency to a spectrum of contexts beyond these instances.

In drawing conclusions from this thesis, it is evident that numerous avenues for future research and development in the domain of HQC decoding and post-quantum cryptography have been unveiled. While our proposed alternative decoder marks a significant advancement, there are promising directions to explore further.

One immediate avenue for future work is the standardization and optimization of the code used for simulations. To ensure maximum efficiency, it would be beneficial to unify the code into a single language, such as the highly-optimized C language. This consolidation would streamline the simulation process and allow for more precise performance measurements, enabling a better understanding of the proposed decoder's practicality and scalability.

Exploring potential future developments in the context of decoding systems, it's worth considering a more comprehensive analysis of the DFR for the new decoder. This analysis may involve efforts to establish a mathematical upper bound for the DFR. This in-depth examination could lead to a better grasp of the decoder's robustness and reliability, potentially paving the way for further improvements in its design and application. Such attempts could significantly enhance the performance and applicability of the decoder in various

domains, including practical scenarios in which IND-CCA2 is required.

As we progress toward practical applications of HQC and post-quantum cryptography, an important avenue for future research is the integration of the proposed decoder into real-world cryptographic systems. Investigating the feasibility and security of such integration, and exploring potential use cases, would be a valuable step towards making post-quantum cryptographic solutions more accessible and effective.

In conclusion, while this thesis has made a contribution to the field of post-quantum cryptography by proposing an improved HQC decoder, it's important to recognize that there are still opportunities for further research, refinement, and practical implementation. These potential future works hold the promise of contributing to the ongoing efforts to enhance cryptographic security in the face of evolving threats and emerging quantum technologies.

References

[1] C. Balamurugan et al. "Post-Quantum and Code-Based Cryptography;Some Prospective Research Directions". In: *Cryptography* 5.4 (2021). ISSN: 2410-387X. DOI: 10.3390/cryptography5040038. URL: https://www.mdpi.com/2410-387X/5/4/38.

[2] M. Baldi, P. Santini, and G. Cancellieri. "Post-quantum cryptography based on codes: State of the art and open challenges". In: *2017 AEIT International Annual Conference*. 2017, pp. 1–6. DOI: 10.23919/AEIT.2017.8240549.

[3] D. Moody. "Post-Quantum Cryptography: NIST's Plan for the Future". In: 2016. URL: https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/pqcrypto-2016-presentation.pdf.

[4] P. W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[5] L. K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[6] *National Institute of Standard and Technology (NIST): Post Quantum Cryptography*. URL: https://csrc.nist.gov/projects/post-quantum-cryptography.

[7] *National Institute of Standard and Technology (NIST): PQC Standardization Process, Third Round Candidate Announcement*. 2020. URL: https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement.

[8] A. Cintas Canto et al. "Algorithmic Security is Insufficient: A Comprehensive Survey on Implementation Attacks Haunting Post-Quantum Security". In: May 2023. DOI: 10.36227/techrxiv.23071079.v1.

[9] W. Ryan and S. Lin. In: *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[10] E. Abbe et al. “Reed-Muller Codes”. In: *Foundations and Trends in Communications and Information Theory* 20.1–2 (2023). ISSN: 1567-2190. DOI: 10.1561/0100000123. URL: http://dx.doi.org/10.1561/0100000123.

[11] M. Mitrouli. “Sylvester Hadamard matrices revisited”. In: *Special Matrices* 2 (Feb. 2014). DOI: 10.2478/spma-2014-0013.

[12] F. J. MacWilliams and N. J. A. Sloane. “The Theory of Error-Correcting Codes”. In: 1977. URL: https://api.semanticscholar.org/CorpusID:118260868.

[13] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[14] P.-L. Cayrel et al. “Recent progress in code-based cryptography”. In: *International Conference on Information Security and Assurance*. Springer. 2011, pp. 21–32.

[15] N. Sendrier. “Code-based cryptography: State of the art and perspectives”. In: *IEEE Security & Privacy* 15.4 (2017), pp. 44–50.

[16] Z. Liu, Y. Pan, and T. Xie. “Breaking the hardness assumption and IND-CPA security of HQC submitted to NIST PQC project”. In: *IET Information Security* 14 (May 2020). DOI: 10.1049/iet-ifs.2019.0214.

[17] S. Nicolas. “Decoding one out of many”. In: *International Workshop on Post-Quantum Cryptography* (2011). URL: https://eprint.iacr.org/2011/367.pdf.

[18] C. A. Melchor et al. *Hamming Quasi-Cyclic (HQC)*. Security section: 38-42pp (2021). URL: https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf.

[19] M. Sipser. In: *Introduction to the Theory of Computation*. Thomson Course Technology, 2005, pp. 256–276.

[20] A. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: *Proceedings of the London Mathematical Society* (1936), pp. 230–265.

[21] S. A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing* (1971), pp. 151–158.

[22] M. R. Garey and D. S. Johnson. “The Approximation of Vertex Cover”. In: *Approximation Algorithms for NP-hard Problems* (1976), pp. 46–47.

[23] R. M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations* (1972), pp. 85–103.

[24] E. Berlekamp, R. McEliece, and H. van Tilborg. "On the Inherent Intractability of Certain Coding Problems (Corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386. DOI: 10.1109/TIT.1978.1055873.

[25] A. M. Barg. "Some New NP-Complete Coding Problems". In: *Probl. Peredachi Inf.* (1994).

[26] D. Hofheinz, K. Hövelmanns, and E. Kiltz. *A Modular Analysis of the Fujisaki-Okamoto Transformation.* Cryptology ePrint Archive, Paper 2017/604. 2017. URL: https://eprint.iacr.org/2017/604.

[27] *National Institute of Standard and Technology (NIST): Hash Functions.* URL: https://csrc.nist.gov/Projects/hash-functions.

[28] C. A. Melchor et al. *Hamming Quasi-Cyclic (HQC). First round version.* 2017.

[29] C. A. Melchor et al. *Hamming Quasi-Cyclic (HQC). Second round version.* 2019.

[30] C. A. Melchor et al. *Hamming Quasi-Cyclic (HQC). Third round version.* 2021. URL: https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf.

[31] C. A. Melchor et al. *Hamming Quasi-Cyclic (HQC). Fourth round version.* 2023. URL: https://pqc-hqc.org/doc/hqc-specification_2023-04-30.pdf.

[32] V. Rijmen and J. Daemen. "Advanced Encryption Standard (AES)". In: *National Institute of Standard and Technology (NIST) Department of Commerce, Washington D.C.* Vol. Federal Information Processing Standards Publication (FIPS). NIST FIPS 197-upd1. 2001. DOI: https://doi.org/10.6028/NIST.FIPS.197-upd1.

[33] E. Barker. "Recommendation for Key Management: Part 1 – General". In: *NIST Special Publication 800-57 Part 1 Revision 5* (2020). DOI: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

[34] B. Reselman. "The Essentials of Using an Ephemeral Key Under TLS 1.3". In: *The New Stack* (2021). URL: https://thenewstack.io/the-essentials-of-using-an-ephemeral-key-under-tls-1-3/.

[35] E. Barker et al. "Guide to IPsec VPNs". In: *NIST Special Publication 800-77 Revision 1* (2020). DOI: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-77r1.pdf.

[36] T. Berger. "Analysis of current VPN technologies". In: *First International Conference on Availability, Reliability and Security (ARES'06).* 2006, 8 pp.–115. DOI: 10.1109/ARES.2006.30.

Acknowledgements

First of all, I would like to express my deepest gratitude to Professor Paolo Santini, who not only ignited my deep passion for this fascinating topic but also dedicated considerable time and effort to mentor me. His guidance, unwavering support, and invaluable assistance were crucial throughout the entire thesis writing process and beyond.

Professor Marco Baldi and Mr. Sebastian Bitzer also deserve my heartfelt thanks for their guidance, insightful suggestions, and the generous time they invested in me.

Un ringraziamento speciale va alla mia famiglia, a mia madre Carla e a mio padre Nunzio, per il sostegno e per le opportunità che mi avete sempre garantito con tanti sacrifici.

Desidero esprimere la mia gratitudine ai miei nonni Ivo e Onorina. Inoltre, vorrei ringraziare anche i miei nonni Fosca e Marcello, che purtroppo non sono più tra noi, ma la loro presenza è eterna nel mio cuore. Con lo sguardo rivolto al cielo, sento che la loro influenza è sempre stata presente, guidandomi silenziosamente attraverso le sfide e le gioie della vita.

Ringrazio tutti i miei amici, in particolare quelli di sempre, Vino e Cristian, con i quali ho condiviso momenti sin dai tempi in cui le tabelline sembravano un mistero indescrivibile. Inoltre, vorrei ringraziare i miei compagni universitari con i quali ho condiviso gioie e dolori di questo tumultuoso percorso.

Infine, un grazie speciale va a Veronica, la persona che più di tutte è stata capace di capirmi e di sostenermi nei momenti difficili. Non ho parole per esprimere quanto ti sia grato, tu sai quello che sei per me e i grazie che ti devo. Grazie per aver creduto in me più di quanto potessi fare io, grazie per esserci sempre e per esserci sempre stata, grazie per avermi sempre capito, sostenuto e tranquillizzato nei momenti più difficili. Seppure è vero che nella mia vita nessun colore avrà tinte più forti del bianco e del nero, sei stata sprazzi di colore nel grigio della mia quotidianità.