HARE: Compact HQC via Distance-Informed Erasure Decoding

Tianrui Wang[1], Qicheng Teng[2], Anyu Wang[2(✉)], Jun Zhang[3(✉)], Bo Pang[4], Chunhuan Zhao[5], Sihuang Hu[6], and Xiaoyun Wang[2]

[1] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
wangtr22@mails.tsinghua.edu.cn,
[2] Institute for Advanced Study, Tsinghua University, Beijing, China
tqc21@mails.tsinghua.edu.cn, anyuwang@tsinghua.edu.cn,
xiaoyunwang@tsinghua.edu.cn
[3] School of Mathematical Sciences, Capital Normal University, Beijing, China
junz@cnu.edu.cn
[4] ICBC Software Development Center, Zhuhai, China
pangb21@sdc.icbc.com.cn
[5] Shield Lab, Huawei Technologies, Beijing, China
zhaochunhuan@huawei.com
[6] School of Cyber Science and Technology, Shandong University, Qingdao, China
husihuang@gmail.com

Abstract. We present HARE, a key encapsulation mechanism (KEM) scheme based on the Hamming Quasi-Cyclic (HQC) framework that achieves substantially reduced public key and ciphertext sizes. HARE focuses on two complementary techniques: a novel distance-informed erasure decoding method for the concatenated code and ciphertext compression based on covering codes.

The distance-informed erasure decoding method uses distance information from the inner Reed-Muller decoder to identify unreliable local blocks and treats them as erasures for the outer Reed-Solomon decoder. This improves decoding performance and enables more compact code parameters. For ciphertext compression, HARE adopts a covering-code perspective and maps the ciphertext vector to a nearby codeword with a shorter representation. The use of the Kaikkonen-Rosendahl code provides a more favorable bandwidth trade-off than previously considered Hamming and Golay code constructions.

Combining these techniques, HARE reduces the combined size of the public key and ciphertext by 15.4%, 14.8%, and 15.3% at NIST security categories 1, 3, and 5, respectively, compared with HQC.

Keywords: Post-Quantum Cryptography · Code-Based Cryptography · Hamming Quasi-Cyclic · HARE · Erasure Decoding · Covering Codes

1 Introduction

The development of quantum algorithms such as Shor's algorithm [34] has motivated the transition to post-quantum cryptography (PQC) and the National

Institute of Standards and Technology (NIST) standardization process [1]. Code-based cryptography is a major approach to PQC, with security rooted in decoding problems studied since the original McEliece proposal [27]. Quasi-cyclic key encapsulation mechanisms (KEMs), including Bit Flipping Key Encapsulation (BIKE) [4] and Hamming Quasi-Cyclic (HQC) [16], reduce the public-key cost of classical code-based systems by representing circulant matrices compactly.

For BIKE's iterative decoder, the decryption failure rate (DFR) is estimated experimentally and extrapolated to smaller probabilities [36,33,37]. For HQC, a conservative DFR bound is derived from an explicit model of the decoding noise [2]. HQC was selected by NIST for standardization, and its analytically tractable DFR model is one of its distinguishing advantages.

HQC separates the code structure used for security from the code used for error correction. Its security is based on quasi-cyclic syndrome decoding (QCSD) problems, while an independent public code recovers the encoded message from the encryption noise. Since this public code contains no secret structure and is not part of the security assumption, it can be replaced or decoded more effectively without changing the QCSD-based security foundation. HQC therefore allows decoding to be optimized independently of computational security.

However, the structure of HQC introduces decryption noise with a relatively large Hamming weight. In particular, the public code must correct an error of the form

$$\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e},$$

where the products of sparse quasi-cyclic vectors can have substantially larger Hamming weight than their inputs. The security bounds obtained through the Fujisaki-Okamoto (FO) transformation and the Hofheinz-Hövelmanns-Kiltz (HHK) framework account for the DFR, which must remain negligible at the target security level [15,21]. Correcting this equivalent noise requires a sufficiently long concatenated code. A long concatenated code therefore leads to a large n, which directly increases both the public-key and ciphertext sizes. As a result, HQC has a substantially larger communication cost than lattice-based KEMs such as Kyber [10].

Accordingly, we aim to reduce the public key and ciphertext sizes of HQC in two ways. Distance-informed erasure decoding reduces the failure probability of the concatenated decoder, allowing shorter code parameters and a smaller ring dimension. Covering code-based compression directly shortens the second ciphertext component at the cost of additional decoding noise. These two mechanisms are tightly coupled: the amount of compression that can be tolerated depends on the erasure-decoding threshold, which controls the trade-off between errors and erasures passed to the outer Reed-Solomon (RS) decoder. The available concatenated code parameters are discrete, so an improvement in decoding performance may not by itself permit a shorter code or a smaller ring dimension. When combined with compression, however, it may enable a smaller overall public-key and ciphertext size that neither technique achieves alone. We therefore jointly select the concatenated-code parameters, erasure threshold, and compression parameters under the target security and DFR constraints.

1.1 Our Contributions

In this work, we present **HARE**, an HQC-based KEM that retains the QCSD-based design of HQC while reducing the size of its public key and ciphertext. HARE combines a distance-informed erasure decoder for the concatenated code with covering code-based ciphertext compression, and selects their parameters under the target security and DFR requirements. Compared with the corresponding HQC parameter sets, the resulting HARE parameter sets reduce the combined public-key and ciphertext size by **15.4%**, **14.8%**, and **15.3%** at NIST security categories 1, 3, and 5, respectively.

Distance-Informed Erasure Decoding. HQC encodes each message using an RS outer code concatenated with duplicated Reed-Muller (RM) inner codes. During decoding, the Reed-Muller decoder processes each binary block and outputs a candidate RS symbol. The outer RS decoder then corrects the symbol errors that remain after the inner decoding stage.

Our key observation is that the Reed-Muller decoding stage provides more information than a candidate codeword alone. In particular, HARE extracts distance information from the Reed-Muller decoding process to assess the confidence of each decoded block. Instead of passing low-confidence blocks to the outer decoder as ordinary symbols, HARE marks them as erasures and uses an error-and-erasure Reed-Solomon decoder to recover the message.

This strategy converts ambiguous inner-code decisions into erasures, which are less costly than erroneous symbols for the outer Reed-Solomon decoder. It lowers the DFR for fixed code parameters, enabling a shorter concatenated code and a smaller ring dimension under the same DFR target.

Covering-Code-Based Compression. Ciphertext compression for HQC was previously introduced by Bitzer et al. [8], using Hamming and Golay codes to map a ciphertext vector to a nearby codeword with a shorter representation. Our contribution is to reinterpret this mechanism from the perspective of covering codes. For compression, the essential requirement is that every input vector be close to some representable codeword, rather than that the code support unique error correction. This viewpoint broadens the available design space beyond perfect codes. Based on this formulation, we investigate the covering codes considered in [12] and identify the Kaikkonen-Rosendahl (KR) code as a suitable choice for HARE. It provides a more favorable trade-off between ciphertext size and compression-induced noise than the Hamming and Golay codes considered in prior work.

Parameter Design and DFR Analysis. To support the two main techniques, HARE uses different weights for secret and error vectors, and accounts for the resulting distribution information in the security analysis. For the DFR evaluation, we follow the hypergeometric model of [3,8] and extend it to the error-and-erasure decoding and ciphertext compression used in HARE. These analyses support the selection of the final HARE parameters.

1.2 Related Work

Prior work on optimizing HQC falls into three main categories.

Ciphertext Compression. Ciphertext compression is widely used in lattice-based schemes such as Kyber, where coefficients are rounded from a larger modulus to a smaller one [10,28,30]. Bitzer et al. [8] adapt this idea to the Hamming metric by mapping each ciphertext block to a nearby codeword and transmitting its shorter message representation. Their constructions use Hamming and Golay codes, trading a controlled amount of additional decoding noise for a shorter ciphertext. HARE extends this approach to general covering codes and uses the KR code to improve the resulting bandwidth trade-off.

Improved Decoders. HQC adopts a structure that separates the security assumption from the decoding performance, allowing the decoding algorithm to be optimized independently without affecting the underlying security guarantee. Therefore, another direction is to design more efficient decoders for HQC. Earlier versions of HQC used a concatenation of repetition codes and Bose-Chaudhuri-Hocquenghem (BCH) codes. These were later replaced by duplicated RM inner codes concatenated with an RS outer code. In [9], the authors propose a two-level generalized concatenated code and develop a reliability-based decoding framework for the outer code. In [5], the authors further observe that the structure of the secret key can be leveraged to assist decoding. However, this low-density parity-check (LDPC) decoder does not retain the closed-form DFR analysis available for the standard HQC decoder. HARE retains the RM-RS concatenated code and uses the distance information produced by the inner RM decoder to mark unreliable outer symbols as erasures.

Refined DFR Models. The original analysis uses an independent Bernoulli approximation for the coordinates of quasi-cyclic products [2]. More recent works replace this approximation with a hypergeometric model that more closely matches the observed weight distributions [3,8]. HARE extends this framework to account for error-and-erasure decoding and covering-code-based ciphertext compression.

Comparison. Table 1 compares the KEM bandwidth reductions achieved by HARE and other HQC optimization works under comparable DFR targets.[1] The compared works reduce bandwidth by at most 4.32% relative to HQC, whereas HARE achieves approximately 15% across all three security categories. This corresponds to a further reduction of more than 11.5% over the smallest parameter sets included in the comparison.

1.3 Organization

Section 2 introduces the required background. Section 3 and Section 4 present the distance-informed erasure decoder and covering-code-based compression, respectively. Section 5 specifies HARE, while Section 6 and Section 7 analyze its

[1] We also noticed the UHQC parameter sets from [18]. That work adopts an attack-aware security criterion permitting DFR bounds substantially larger than $2^{-\lambda}$. Thus, we omit these parameter sets to retain comparable DFR requirements.

Instance	[8]	[9]	[3]	HARE
HQC-1	3.57%	4.32%	2.49%	**15.36%**
HQC-3	2.96%	3.53%	1.29%	**14.82%**
HQC-5	3.15%	3.90%	1.60%	**15.26%**

Table 1: KEM bandwidth reduction, measured by the combined encapsulation-key and ciphertext size, relative to HQC. The baseline sizes are 6674, 13492, and 21658 bytes, with DFR targets of 2^{-128}, 2^{-192}, and 2^{-256}, respectively.

security and DFR. Section 8 reports the implementation results. Section 9 concludes the paper and discusses future work.

2 Preliminary

2.1 Notation

Let $\mathbb{Z}$ denote the ring of integers, and let $\mathbb{F}_q$ denote the finite field with q elements. Let $\mathbb{F}_q^n$ denote the n-dimensional vector space over $\mathbb{F}_q$. Let n be a primitive prime, meaning that n is prime and 2 is a primitive root modulo n. We define the quotient ring $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$. Vectors are generally denoted by bold lowercase letters, e.g., $\mathbf{v}$, while matrices are denoted by uppercase letters, e.g., $\mathbf{G}$. Unless otherwise specified, all vectors are treated as row vectors. Given $\mathbf{v} \in \mathbb{F}_2^n$, let $\mathtt{rot}(\mathbf{v})$ denote the circulant matrix generated by $\mathbf{v}$, i.e.,

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

Since there is a natural one-to-one correspondence between the elements and operations in $\mathbb{F}_2^n$ and the polynomial ring $\mathcal{R}$, we will use vectors in $\mathbb{F}_2^n$ and polynomials in $\mathcal{R}$ interchangeably. Specifically, for $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1}) \in \mathbb{F}_2^n$, the corresponding element is $\mathbf{v} = v_0 + v_1 X + \cdots + v_{n-1} X^{n-1} \in \mathcal{R}$. The product of $\mathbf{u}, \mathbf{v} \in \mathcal{R}$ is $\mathbf{w} = \mathbf{uv}$ with

$$w_k = \sum_{i+j \equiv k \bmod n} u_i v_j, \ \forall k \in \{0, 1, \cdots, n-1\},$$

which can be expressed as the product of $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n$ with the operator $\mathtt{rot}(\cdot)$ as:

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u} \times \mathtt{rot}(\mathbf{v})^\top = (\mathtt{rot}(\mathbf{u}) \times \mathbf{v}^\top)^\top = \mathbf{v} \times \mathtt{rot}(\mathbf{u})^\top = \mathbf{v} \cdot \mathbf{u}.$$

Let $w_H(\mathbf{v})$ denote the Hamming weight of the vector $\mathbf{v}$, i.e., the number of nonzero positions of $\mathbf{v}$. Let $\mathcal{R}_w := \{\mathbf{v} \in \mathcal{R} \mid w_H(\mathbf{v}) = w\}$. For $\mathbf{v} = (v_0, \cdots, v_{n-1}) \in \mathbb{F}_q^n$, we define $\mathsf{Truncate}(\mathbf{v}, l) := (v_0, \cdots, v_{l-1})$, which returns the first l coordinates of $\mathbf{v}$. Let $\mathcal{B} := \{0,1\}^8$ denote the set of bytes, and let $\mathcal{B}^\ell$ denote the set of byte strings of length ℓ.

2.2 Coding Theory

Definition 1 (Linear Code). *A q-ary linear code $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$. If $\mathcal{C}$ has dimension k, it is called an $[n,k]$ or an $[n,k]_q$ code. The code $\mathcal{C}$ can be equivalently described in the following ways:*

(i) $\mathcal{C} = \{\mathbf{mG} \mid \mathbf{m} \in \mathbb{F}_q^k\}$, *where* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ *is called a generator matrix.*

(ii) $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n \mid \mathbf{cH}^\top = \mathbf{0}\}$, *where* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *is called a parity check matrix.*

Definition 2 (Minimum Distance). *The minimum distance of a code $\mathcal{C}$ is*

$$\min\{w_H(\mathbf{x}-\mathbf{y}) \mid \mathbf{x} \in \mathcal{C}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\}$$

If the minimum distance of $\mathcal{C}$ is d, then we also describe it as a $[n,k,d]$ code.

Definition 3 (Perfect Code). *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ have minimum distance d, and let $t = \lfloor (d-1)/2 \rfloor$. The code is called* perfect *if the Hamming balls of radius t centered at its codewords partition $\mathbb{F}_q^n$. Equivalently, its covering radius, the maximum distance from a vector in $\mathbb{F}_q^n$ to its nearest codeword, equals t.*

There are three kinds of perfect binary linear codes: Hamming codes, one Golay code, and trivial codes [24].

Definition 4 (Hamming Codes). *A Hamming code over $\mathbb{F}_2$ has parameters $[2^m-1, 2^m-m-1, 3]_2$ for any integer $m \geq 2$.*

Definition 5 (Binary Golay Code). *The perfect binary Golay code has parameters $[23, 12, 7]_2$.*

Definition 6 (Concatenated Codes). *A concatenated code over $\mathbb{F}_2$ consists of an external/outer code $\mathcal{C}_e$ with parameters $[n_e, k_e, d_e]$ over $\mathbb{F}_q$ and an internal/inner code $\mathcal{C}_i$ with parameters $[n_i, k_i, d_i]$ over $\mathbb{F}_2$, where $q = 2^{k_i}$. Each symbol of an outer codeword is mapped to a message of $\mathcal{C}_i$ and then encoded using the inner code. The resulting concatenated code is a binary code with parameters $[N = n_e n_i,\ K = k_e k_i,\ D \geq d_e d_i]$.*

2.3 Security Definitions

We recall indistinguishability under chosen-plaintext attacks (IND-CPA) for public-key encryption (PKE) schemes and indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA2) for KEM schemes. The corresponding security games and advantages are defined below.

Definition 7. *Let* $\mathsf{PKE} = (\mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme. For a probabilistic polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_{\mathsf{CHOOSE}}, \mathcal{A}_{\mathsf{GUESS}})$*, the* IND-CPA *advantage is defined as*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = |\Pr[\mathsf{PKE.IND\text{-}CPA}(\mathcal{A}) = 1] - \frac{1}{2}|,$$

where the IND-CPA *game is defined as follows.*

(IND-CPA *Game*)

1. $(\mathbf{ek}, \mathbf{dk}) \leftarrow \mathsf{Keygen}()$
2. $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{\mathsf{CHOOSE}}(\mathbf{ek})$
3. $b \xleftarrow{\$} \{0,1\}$, $\theta \xleftarrow{\$} \mathcal{B}^{|\theta|}$
4. $\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{Encrypt}(\mathbf{ek}, \mathbf{m}_b, \theta)$
5. $b' \leftarrow \mathcal{A}_{\mathsf{GUESS}}(\mathbf{ek}, \mathbf{c}_{\mathsf{PKE}})$
6. return $(b = b')$

Definition 8. *Let* $\mathsf{KEM} = (\mathsf{Keygen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be a key encapsulation mechanism. For a probabilistic polynomial-time adversary* $\mathcal{A} = \mathcal{A}_{\mathsf{GUESS}}$ *with access to the decapsulation oracle* DECAPS*, the* IND-CCA2 *advantage is defined as*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\mathcal{A}) = |\Pr[\mathsf{KEM.IND\text{-}CCA2}(\mathcal{A}) = 1] - \frac{1}{2}|,$$

where the IND-CCA2 *game is defined as follows.*

(IND-CCA2 *Game*)

1. $(\mathbf{ek}, \mathbf{dk}) \leftarrow \mathsf{Keygen}()$
2. $b \xleftarrow{\$} \{0,1\}$
3. $(\mathbf{K}_0, \mathbf{c}_{\mathsf{KEM}}) \leftarrow \mathsf{Encaps}(\mathbf{ek})$
4. $\mathbf{K}_1 \xleftarrow{\$} \mathcal{B}^{|\mathbf{K}|}$
5. $b' \leftarrow \mathcal{A}^{\mathsf{DECAPS}(\cdot)}_{\mathsf{GUESS}}(\mathbf{K}_b, \mathbf{c}_{\mathsf{KEM}})$ *where the adversary is not allowed to query* $\mathbf{c}_{\mathsf{KEM}}$ *to the decapsulation oracle*
6. return $(b = b')$

2.4 HQC Scheme

HQC-PKE [16] combines a quasi-cyclic code for computational security with an independent public error-correcting code $\mathcal{C}$ for decryption correctness. For this overview, we ignore the small length difference between codewords of $\mathcal{C}$ and elements of $\mathcal{R}$, and implicitly embed each codeword into $\mathcal{R}$. Over the ring $\mathcal{R}$, key generation samples sparse secret vectors $\mathbf{x}, \mathbf{y}$ and a uniform public vector $\mathbf{h}$, then computes $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$ to form the public key $(\mathbf{h}, \mathbf{s})$. To encrypt a message $\mathbf{m}$, the sender encodes it as $\mathbf{c} = \mathbf{mG} \in \mathcal{C}$, samples fresh sparse vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}$, and outputs $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ and $\mathbf{v} = \mathbf{c} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$. Decryption computes $\mathbf{v} - \mathbf{u} \cdot \mathbf{y} = \mathbf{c} + \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$ and applies the decoder of $\mathcal{C}$ to recover $\mathbf{m}$ whenever the equivalent error lies within its decoding capability. HQC-KEM is obtained by applying the FO transformation [15] to HQC-PKE within the HHK framework with implicit rejection [17].

3 Erasure Decoding

In this section, we introduce how we construct the concatenated RM and RS codes and our erasure decoding algorithm.

3.1 Distance-Informed Erasure Decoding

In the standard HQC scheme, the RM decoder performs hard-decision decoding and forwards only the decoded symbol to the RS decoder, discarding the distance information available during inner decoding. A more general approach would be to incorporate this information directly into a soft or reliability-aware RS decoding procedure. However, accurately characterizing the DFR under such a decoding strategy is difficult.

HARE therefore uses the distance information in a more restricted form. Rather than exposing the full reliability information to the outer decoder, the RM stage outputs either a decoded symbol or an erasure. For the subsequent analysis, each outcome is classified as correct, erroneous, or erased. This formulation incorporates part of the reliability information from RM decoding while allowing the subsequent DFR analysis to remain tractable using conservative counting. The complete decoding procedure and the corresponding DFR analysis are presented later.

3.2 Duplicated Reed-Muller Codes

We propose a modified decoding algorithm, $\mathsf{Decode}'()$, for duplicated Reed-Muller codes, as described in Algorithm 1. The algorithm outputs either a message $\mathbf{m}$ or an erasure symbol $\perp$, depending on the confidence of the decoding result.

The main modification compared to the hard-decision decoding in [16] is as follows. Instead of directly decoding to the nearest codeword, we first identify the two codewords with the smallest Hamming distances to the received vector. If the difference between these two distances exceeds a threshold α, the vector is decoded to the nearest codeword. Otherwise, the decoder outputs an erasure symbol $\perp$, indicating insufficient confidence. This approach reduces the probability of incorrect decoding by explicitly marking uncertain positions as erasures, which can then be handled in the subsequent Reed-Solomon decoding stage.

For context, a Reed-Muller code $\mathrm{RM}(r, m)$ with positive integer m and $0 \leq r \leq m$ has parameters $[n, k, d_{\min}] = \left[2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}\right]$, and duplicated RM codes are constructed by repeating each codeword multiple times [24]. Encoding is performed by multiplying the message with the generator matrix.

We next construct conservative outcome counts for weight-w error patterns in the duplicated RM code. When these patterns are sampled uniformly, division by $\binom{n}{w}$ gives the normalized quantities used in the subsequent RS failure analysis.

Theorem 1. *Let $\mathcal{C}$ be a duplicated RM code obtained by repeating the codewords of the* $\mathrm{RM}(1, 7)$ *code with parameters* $[128, 8, 64]$*, and let $[n, k, d]$ denote the parameters after duplication. Assume that the received word contains exactly w*

Algorithm 1 Decode' for a duplicated RM code

Input: Vector $\mathbf{x}$ to be corrected
Parameter: A duplicated RM code of $[n_i, k_i, d_i]$ over $\mathbb{F}_2$ with generator matrix $\mathbf{G}$, threshold α
Output: Message $\mathbf{m}$ or $\perp$

$\mathbf{m}_1, \mathbf{m}_2 \leftarrow \mathbf{0}, \mathbf{0}$
Construct $\mathscr{D} \leftarrow \{(d, \mathbf{m}) \mid \mathbf{m} \in \mathbb{F}_2^{k_i},\ d = \mathrm{w}_H(\mathbf{mG} - \mathbf{x})\}$ using the Hadamard transform
Sort $\mathscr{D}$ in ascending order of d
Let $(d_1, \mathbf{m}_1)$ and $(d_2, \mathbf{m}_2)$ be the first two elements of $\mathscr{D}$
if $d_2 - d_1 > \alpha$ **then**
    return $\mathbf{m}_1$
else
    return $\perp$
end if

errors and that $\alpha \geq 0$ *is the decoding threshold. Define the conservative counts* [2]

$$\mathrm{N}_{\mathsf{error},w} = \min\left[254 \sum_{2j-d>\alpha} \binom{d}{j}\binom{n-d}{w-j} + \mathbf{1}_{\{2w-2d>\alpha\}}\binom{n}{w}, \binom{n}{w}\right],$$

$$\mathrm{N}_{\mathsf{erasure},w} = \min\Bigg[254 \sum_{-\alpha\leq d-2j\leq\alpha} \binom{d}{j}\binom{n-d}{w-j} + \mathbf{1}_{\{-\alpha\leq 2d-2w\leq\alpha\}}\binom{n}{w}, \binom{n}{w} - \mathrm{N}_{\mathsf{error},w}\Bigg],$$

and

$$\mathrm{N}_{\mathsf{correct},w} = \max\left[\binom{n}{w} - \mathrm{N}_{\mathsf{error},w} - \mathrm{N}_{\mathsf{erasure},w}, 0\right].$$

Here, $\mathbf{1}_{\{P\}}$ *equals* 1 *if the condition* P *holds and equals* 0 *otherwise. These counts conservatively estimate the error, erasure, and correct outcomes of* Decode' *for the subsequent RS failure analysis.*

Proof. By the linearity of $\mathcal{C}$, we assume that the original error-free codeword is $\mathbf{c}_0 = \mathbf{0}$. Let $\mathbf{e}$ be a weight-w error vector and let $\mathbf{x} = \mathbf{c}_0 + \mathbf{e}$ be the received word. Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be the nearest and second-nearest codewords to $\mathbf{x}$, with respective distances d_1 and d_2. Let $d_0 = w = \mathrm{w}_H(\mathbf{c}_0 - \mathbf{x})$. By definition, $d_1 \leq d_2$. We distinguish the following four cases.

– If $\mathbf{c}_1 = \mathbf{c}_0$ and $d_2 - d_1 \leq \alpha$, the decoder outputs an **erasure**. Let $d' = \mathrm{w}_H(\mathbf{c}_2 - \mathbf{c}_0)$ for a possible nonzero codeword $\mathbf{c}_2$. For fixed w and $\mathbf{c}_2$, let $\mathsf{A}^w_{\mathbf{c}_2}$

[2] We do not refer to these quantities as upper or lower bounds in the theorem, since some erasures may be treated as errors to obtain a conservative estimate of the DFR.

denote the pairwise event $0 \le \mathrm{w}_H(\mathbf{x}-\mathbf{c}_2) - \mathrm{w}_H(\mathbf{x}-\mathbf{c}_0) \le \alpha$. If j is the size of the intersection between the supports of $\mathbf{c}_2 - \mathbf{c}_0$ and $\mathbf{e}$, then the pairwise distance difference is $d' - 2j$. The number of corresponding weight-w error patterns is

$$\mathrm{N}(\mathsf{A}^w_{\mathbf{c}_2}) = \sum_{0 \le d'-2j \le \alpha} \binom{d'}{j}\binom{n-d'}{w-j}.$$

The RM$(1,7)$ code has 255 nonzero codewords, of which 254 have weight 64 and one has weight 128. After duplication, the corresponding weights are d and $2d = n$. Every erasure in this case belongs to at least one such pairwise event. For each of the 254 weight-d codewords, the pairwise contribution is

$$\sum_{0 \le d-2j \le \alpha} \binom{d}{j}\binom{n-d}{w-j}.$$

The unique weight-$2d$ codeword is the all-one vector, whose support intersects every weight-w error pattern in exactly w positions. Its pairwise distance difference is therefore $2d - 2w$, and its contribution is

$$\mathbf{1}_{\{0 \le 2d-2w \le \alpha\}} \binom{n}{w}.$$

Thus, the union bound over all nonzero codewords gives an upper bound on the number of erasure patterns:

$$254 \sum_{0 \le d-2j \le \alpha} \binom{d}{j}\binom{n-d}{w-j} + \mathbf{1}_{\{0 \le 2d-2w \le \alpha\}} \binom{n}{w}.$$

- If $\mathbf{c}_1 = \mathbf{c}_0$ and $d_2 - d_1 > \alpha$, the decoder outputs the **correct** codeword.
- If $\mathbf{c}_1 \neq \mathbf{c}_0$ and $w - d_1 \le \alpha$, then $\mathbf{c}_0$ is a candidate distinct from $\mathbf{c}_1$. Since $\mathbf{c}_2$ is the nearest candidate other than $\mathbf{c}_1$, its distance satisfies $d_2 \le d_0 = w$. It follows that $d_2 - d_1 \le w - d_1 \le \alpha$, so the decoder outputs an **erasure**. For fixed $\mathbf{c}_1$, define d' and j as in the first case. Since $d_1 - w = d' - 2j \le 0$, the condition $w - d_1 \le \alpha$ gives $-\alpha \le d' - 2j \le 0$. Applying the same counting argument with $\mathbf{c}_1$ as the fixed nonzero codeword gives

$$254 \sum_{-\alpha \le d-2j \le 0} \binom{d}{j}\binom{n-d}{w-j} + \mathbf{1}_{\{-\alpha \le 2d-2w \le 0\}} \binom{n}{w}.$$

- If $\mathbf{c}_1 \neq \mathbf{c}_0$ and $w - d_1 > \alpha$, the decoder may output either an incorrect codeword or an erasure. We conservatively classify every such pattern as a decoding **error**. In this case, $w - d_1 = 2j - d' > \alpha$. Applying the same counting argument gives

$$254 \sum_{2j-d > \alpha} \binom{d}{j}\binom{n-d}{w-j} + \mathbf{1}_{\{2w-2d > \alpha\}} \binom{n}{w}.$$

An error has a greater adverse effect than an erasure on the subsequent RS decoding. Thus, this reclassification can only increase the estimated RS failure probability.

In the first case, every erasure is covered by a pairwise event satisfying $0 \leq d' - 2j \leq \alpha$, while in the third case every erasure is covered by a pairwise event satisfying $-\alpha \leq d' - 2j \leq 0$. Although these ranges overlap when $d' - 2j = 0$, their union is $-\alpha \leq d' - 2j \leq \alpha$. Applying a single union bound to this combined event gives the erasure range $-\alpha \leq d - 2j \leq \alpha$ for the weight-d codewords and $-\alpha \leq 2d - 2w \leq \alpha$ for the unique weight-$2d$ codeword used in the theorem statement. The union-bound sums may count the same weight-w pattern more than once, while there are only $\binom{n}{w}$ such patterns. We therefore cap the error count at $\binom{n}{w}$ and then cap the erasure count at the number of patterns remaining after the conservative error allocation. This ensures that $\mathrm{N}_{\mathsf{error},w} + \mathrm{N}_{\mathsf{erasure},w} \leq \binom{n}{w}$. If the latter cap reduces the erasure count, the reduction is no greater than the excess already assigned to the error count. Since errors have a worse effect than erasures on the subsequent RS decoding, this allocation gives a conservative upper bound on the DFR. Finally, subtracting the allocated error and erasure counts from $\binom{n}{w}$ and taking the maximum with zero gives the expression for $\mathrm{N}_{\mathsf{correct},w}$ in the theorem statement. □

We also evaluate these conservative estimates for the concrete parameters through simulation. In particular, we compare the theoretical error and non-correct rates with the empirical rates. The experimental setup and results are reported in Appendix A.

3.3 Reed-Solomon Codes with Erasure Decoding

When decoding shortened Reed-Solomon codes, we employ a modified procedure that supports erasure decoding. We omit the construction and decoding details of Reed-Solomon codes here and provide the full procedures in Appendix B.

Theorem 2. *Let Γ_w denote an upper bound on the number of weight-w binary error patterns for which the RS$[n_e, k_e, d_e]$ stage fails.*

(a) *Every failure with g errors and h erasures satisfies $2g + h > n_e - k_e$, while the converse need not hold.*

(b) *Fix w, for every $0 \leq g \leq n_e$, $0 \leq h \leq n_e - g$, and every fixed pair of disjoint sets of g error-symbol positions and h erasure-symbol positions, assume that the number of corresponding weight-w binary patterns is at most*

$$\mathcal{N}(g,h,w) = \sum_{\substack{0 \leq w_1,\ldots,w_{n_e} \leq n_i \\ w_1+\cdots+w_{n_e}=w}} \left(\prod_{i=1}^{g} \mathrm{N}_{\mathsf{error},w_i} \right) \left(\prod_{i=g+1}^{g+h} \mathrm{N}_{\mathsf{erasure},w_i} \right) \cdot \left(\prod_{i=g+h+1}^{n_e} \mathrm{N}_{\mathsf{correct},w_i} \right), \tag{1}$$

Then

$$\Gamma_w := \sum_{\substack{0 \leq g \leq n_e,\ 0 \leq h \leq n_e - g \\ 2g+h > n_e - k_e}} \binom{n_e}{g} \binom{n_e - g}{h} \mathcal{N}(g,h,w). \tag{2}$$

Proof. For part (a), our decoder recovers the unknown-error locator and every discrepancy whenever $n_e - k_e - h \geq 2g$. Equivalently, every pattern satisfying $2g + h \leq n_e - k_e = d_e - 1$ is corrected, so every failure lies outside this region.

For part (b), partition the actual failure set by (g, h) and by the two disjoint sets of error and erasure symbol positions. Part (a) restricts the sum to $2g + h > n_e - k_e$. There are $\binom{n_e}{g}\binom{n_e - g}{h}$ choices of these sets, and the stated assumption bounds each fixed choice by $\mathcal{N}(g, h, w)$. Summing the bounds proves (2). □

4 Compression with Covering Codes

In [8], the authors propose a ciphertext compression technique based on a quantization code with a direct product structure, focusing on Hamming and Golay codes. From the perspective of error-correcting codes, a code with error-correcting capability t guarantees correct decoding when the error has Hamming weight at most t. However, this does not mean that every vector in $\mathbb{F}_q^n$ is within Hamming distance t of a codeword. If we require every vector to be decoded while keeping the error weight at most t, the Hamming balls of radius t centered at the codewords must cover the entire space. Since these balls are disjoint for a t-error-correcting code, they must cover the space exactly. Codes with this property are known as perfect codes in Definition 3. The class of non-trivial binary linear perfect codes is extremely limited, consisting only of Hamming codes and the binary Golay code. This restriction significantly narrows the design space if one relies solely on traditional error-correcting codes for compression.

In contrast, from the covering-code perspective, the Hamming balls used to cover the whole space may overlap, which significantly enlarges the range of admissible constructions. A wide variety of covering codes with diverse parameters has been proposed in [12]. Although the covering balls of these codes need not be disjoint, the additional flexibility in their parameters can be used to better balance compression efficiency and compression-induced noise. This observation motivates the exploration of more general covering codes for ciphertext compression, beyond the classical perfect codes used in [8].

4.1 Covering Codes

We first recall the definition of a linear covering code.

Definition 9 (Covering Code). *A linear $[n, k]_q$ code $\mathcal{C}$ has covering radius R if any of the following equivalent conditions holds:*

(i) The value R is the smallest integer such that the space $\mathbb{F}_q^n$ is covered by Hamming balls of radius R centered at the codewords of $\mathcal{C}$.

(ii) Every vector in $\mathbb{F}_q^{n-k}$ can be expressed as a linear combination of at most R columns of the parity check matrix of $\mathcal{C}$, and R is the smallest value with this property.

Let $\mathcal{C}$ be a binary linear $[n,k]_2$ code with generator matrix $\mathbf{G}$, parity check matrix $\mathbf{H}$, and covering radius R. We define the compression and decompression interfaces by

$$\mathsf{Compress}_{\mathcal{C}} : \mathbb{F}_2^n \to \mathbb{F}_2^k \quad \text{and} \quad \mathsf{Decompress}_{\mathcal{C}}(\mathbf{m}) = \mathbf{mG}.$$

For every $\mathbf{v} \in \mathbb{F}_2^n$, these interfaces satisfy

$$w_H(\mathbf{v} - \mathsf{Decompress}_{\mathcal{C}}\big(\mathsf{Compress}_{\mathcal{C}}(\mathbf{v})\big)) \leq R.$$

When R is small, $\mathsf{Compress}_{\mathcal{C}}$ can be implemented using a precomputed syndrome table, similarly to the table-based syndrome decoding technique in [19]. For a codeword $\mathbf{c} \in \mathcal{C}$, let $\mathsf{Message}_{\mathcal{C}}(\mathbf{c})$ denote the unique message $\mathbf{m} \in \mathbb{F}_2^k$ such that $\mathbf{c} = \mathbf{mG}$. The algorithm is detailed in Algorithm 2.

Algorithm 2 Syndrome-Table Compression

Input: Parity check matrix $\mathbf{H}$, input vector $\mathbf{v}$
Parameter: Covering radius R
Output: Compressed representation $\mathbf{m}$

Initialize an empty table $\mathscr{T}$
for all $\mathbf{e} \in \mathbb{F}_2^n$ with $w_H(\mathbf{e}) \leq R$ **do**
    $\mathbf{s} \leftarrow \mathbf{He}^T$
    if $\mathbf{s} \notin \mathscr{T}$ **or** $w_H(\mathbf{e}) < w_H(\mathscr{T}[\mathbf{s}])$ **then**
        $\mathscr{T}[\mathbf{s}] \leftarrow \mathbf{e}$
    end if
end for
$\mathbf{s}_v \leftarrow \mathbf{Hv}^T$
$\mathbf{e}_v \leftarrow \mathscr{T}[\mathbf{s}_v]$
$\mathbf{c} \leftarrow \mathbf{v} - \mathbf{e}_v$
return $\mathsf{Message}_{\mathcal{C}}(\mathbf{c})$

The covering property guarantees that the table contains an entry of weight at most R for every syndrome. The table can be precomputed offline by enumerating $\sum_{i=0}^{R} \binom{n}{i}$ error vectors. During the online phase, the dominant operation is the computation of $\mathbf{Hv}^T$, which requires $O(n(n-k))$ binary operations.

4.2 Compression Error

We now describe how covering codes are applied to the ciphertext. Let $\mathbf{v} \in \mathbb{F}_2^{l_1}$ and let $\mathcal{C}_{\text{cover}}$ be a binary covering code with parameters $[n_{\text{cc}}, k_{\text{cc}}]_2$ and covering radius R_{cc}. Set

$$B = \left\lfloor \frac{l_1}{n_{\text{cc}}} \right\rfloor \quad \text{and} \quad l_2 = Bn_{\text{cc}}.$$

We write $\mathbf{v} = (\mathbf{v}_1 \| \mathbf{v}_2)$, where

$$\mathbf{v}_1 = (\mathbf{v}_1^{(0)} \| \cdots \| \mathbf{v}_1^{(B-1)}), \qquad \mathbf{v}_1^{(j)} \in \mathbb{F}_2^{n_{\text{cc}}},$$

and $\mathbf{v}_2$ contains the remaining $l_1 - l_2$ bits. For each block, we compute

$$\mathbf{m}^{(j)} = \mathsf{Compress}_{\mathcal{C}_{\text{cover}}}(\mathbf{v}_1^{(j)}),$$

and concatenate the results as

$$\tilde{\mathbf{m}} = (\mathbf{m}^{(0)} \| \cdots \| \mathbf{m}^{(B-1)}).$$

The compressed representation of $\mathbf{v}$ is $(\tilde{\mathbf{m}}, \mathbf{v}_2)$ and has length $Bk_{\text{cc}} + (l_1 - l_2)$ bits.

The following theorem bounds the additional error introduced by decompression.

Theorem 3. *Let $\mathcal{C}_2 = \mathcal{C}_{\text{cover}}^{\times B}$ be the direct product of a binary $[n_{\text{cc}}, k_{\text{cc}}]_2$ covering code with covering radius R_{cc}. For any $\mathbf{v}_1 \in \mathbb{F}_2^{l_2}$, let*

$$\tilde{\mathbf{m}} = \mathsf{Compress}_{\mathcal{C}_2}(\mathbf{v}_1) \quad \textit{and} \quad \tilde{\mathbf{e}} = \big(\mathbf{v}_1 - \mathsf{Decompress}_{\mathcal{C}_2}(\tilde{\mathbf{m}}) \| 0^{n-l_2}\big).$$

Then each block $\tilde{\mathbf{e}}^{(j)}$ satisfies

$$w_H(\tilde{\mathbf{e}}^{(j)}) \le R_{\text{cc}},$$

and consequently

$$w_H(\tilde{\mathbf{e}}) \le BR_{\text{cc}} = \frac{l_2 R_{\text{cc}}}{n_{\text{cc}}}.$$

Proof. For every block $\mathbf{v}_1^{(j)}$, the definition of the covering radius guarantees a codeword at Hamming distance at most R_{cc}. The compression interface selects such a codeword, so the corresponding difference vector $\tilde{\mathbf{e}}^{(j)}$ has Hamming weight at most R_{cc}. Summing this bound over the B disjoint blocks gives $w_H(\tilde{\mathbf{e}}) \le BR_{\text{cc}}$. The equality follows from $l_2 = Bn_{\text{cc}}$. □

In the subsequent decryption failure rate analysis, the contribution of $\tilde{\mathbf{e}}$ is modeled through its Hamming weight.

4.3 Covering Code Selection

To identify a covering code that better fits the parameter regime of HARE, we compare representative covering codes, including those with covering radius 2, 3, and 4 reported in [12] whose block lengths are compatible with HARE, as well as the perfect Hamming and Golay codes adopted in [8]. The evaluated codes and their parameters are summarized in Table 2. We compare these codes in terms of the introduced noise rate R/n and the compression rate k/n, where smaller values of both metrics indicate better performance. As shown in Figure 1, the KR code is located in the lower-left region, suggesting that it achieves a favorable balance between compression efficiency and noise tolerance.

Considering the specific parameter setting of HARE, we further evaluate the performance of different covering codes. Among the evaluated codes, the KR

Code	(n, k, R)
KR	$(51, 41, 2)$
Golay	$(23, 12, 3)$
ham-m	$(2^m - 1, 2^m - 1 - m, 1)$
cov-2-t	$(27 \cdot 2^{t-4} - 1, 27 \cdot 2^{t-4} - 1 - 2t, 2)$
cov-3-1	$(18, 9, 3)$
cov-3-2	$(303, 282, 3)$
cov-4-1	$(19, 8, 4)$
cov-4-2	$(690, 659, 4)$

Table 2: Parameters of covering codes evaluated in Figure 1.

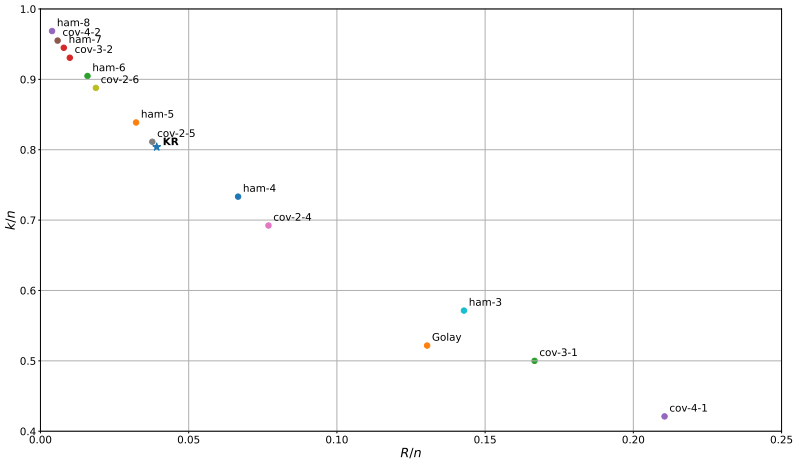

Fig. 1: The compression rate k/n and the additional noise rate R/n for several covering codes. The lower-left region corresponds to better overall performance.

code provides a favorable trade-off for the HARE parameter regime between ciphertext-size reduction and the additional noise introduced by decompression. Therefore, we adopt the KR code as the covering code in HARE. For completeness, we next describe its construction and decoding procedure.

Definition 10 (Kaikkonen-Rosendahl Code). *Let $\mathcal{C}_{KR}$ be the binary $[51, 41]_2$ code with the parity check matrix $\mathbf{H}_{KR} = [\mathbf{I}_{10}\ \mathbf{M}_{KR}]$, where $\mathbf{M}_{KR}$ is the 10×41 binary matrix with the following columns in hexadecimal notation:*

$$\begin{aligned}\mathbf{M}_{KR} = [&\texttt{1B6}, \texttt{193}, \texttt{1CC}, \texttt{187}, \texttt{1F6}, \texttt{0F7}, \texttt{16E}, \texttt{140}, \texttt{03C}, \texttt{296}, \texttt{22F}, \texttt{303}, \texttt{381}, \texttt{365},\\ &\texttt{11D}, \texttt{1A3}, \texttt{274}, \texttt{2F2}, \texttt{254}, \texttt{056}, \texttt{00F}, \texttt{041}, \texttt{357}, \texttt{208}, \texttt{034}, \texttt{329}, \texttt{28D}, \texttt{31D}, \texttt{3D5},\\ &\texttt{129}, \texttt{3D7}, \texttt{0B7}, \texttt{3EC}, \texttt{2E2}, \texttt{23C}, \texttt{0AD}, \texttt{34E}, \texttt{155}, \texttt{2E6}, \texttt{371}, \texttt{0D4}].\end{aligned} \tag{3}$$

It is proven in [22] that $\mathcal{C}_{KR}$ is a $[51, 41]_2$ code with covering radius 2. In HARE, we instantiate $\mathcal{C}_2$ as the direct product $\mathcal{C}_{KR}^{\times B}$, which has length $51B$ and dimension $41B$. Compression is performed block by block using Algorithm 2, while decompression maps each compressed block $\mathbf{m}^{(j)}$ to the codeword $\mathbf{m}^{(j)}\mathbf{G}_{KR}$. By Theorem 3, the compression error has weight at most 2 in each 51-bit block and total weight at most $2B$.

5 HARE Scheme

5.1 Overview of HARE-PKE

Before presenting the formal definition of HARE-PKE, we first describe the main idea and the procedures of the scheme at a high level. Compared to the original HQC scheme [16], the main differences are the distance-informed erasure decoding as described in Section 3 and the covering code-based compression technique as described in Section 4.

- $\mathsf{Keygen}()$: samples $\mathbf{h} \xleftarrow{\$} \mathcal{R}$, $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{R}_w \times \mathcal{R}_w$, sets $\mathbf{sk} = (\mathbf{x}, \mathbf{y})$ and $\mathbf{pk} = (\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$, and returns $(\mathbf{pk}, \mathbf{sk})$
- $\mathsf{Encrypt}(\mathbf{pk}, \mathbf{m})$: generates $\mathbf{e} \xleftarrow{\$} \mathcal{R}_{w_e}, \mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{R}_{w_r} \times \mathcal{R}_{w_r}$, sets $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ and $\mathbf{v} = \mathbf{m}\mathbf{G}_1 + \mathsf{Truncate}(\mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}, l_1)$, splits $\mathbf{v}_1 = \mathbf{v}[0, l_2 - 1]$ and $\mathbf{v}_2 = \mathbf{v}[l_2, l_1 - 1]$, compresses $\tilde{\mathbf{m}} = \mathsf{Compress}_{\mathcal{C}_2}(\mathbf{v}_1)$, returns $\mathbf{c} = (\mathbf{u}, \tilde{\mathbf{m}}, \mathbf{v}_2)$
- $\mathsf{Decrypt}(\mathbf{sk}, \mathbf{c})$: recovers $\tilde{\mathbf{v}} = (\mathsf{Decompress}_{\mathcal{C}_2}(\tilde{\mathbf{m}}) \| \mathbf{v}_2)$, returns $\mathcal{C}_1.\mathsf{Decode}(\tilde{\mathbf{v}} - \mathsf{Truncate}(\mathbf{u} \cdot \mathbf{y}, l_1))$

The matrix $\mathbf{G}_1$ denotes the generator matrix of the code $\mathcal{C}_1$. In HARE, $\mathcal{C}_1$ is the concatenation of a shortened RS outer code with duplicated RM inner codes, while $\mathcal{C}_2 = \mathcal{C}_{KR}^{\times B}$ is a direct-product covering code.

Correctness. For simplicity, we omit truncation and assume all vectors have matching lengths. Let $\tilde{\mathbf{e}} = \mathsf{Decompress}_{\mathcal{C}_2}(\tilde{\mathbf{m}}) - \mathbf{v}_1$ denote the noise introduced in $\mathcal{C}_2$. The decryption can then be written as

$$\begin{aligned}
\tilde{\mathbf{v}} - \mathbf{u}\mathbf{y} &= (\mathsf{Decompress}_{\mathcal{C}_2}(\tilde{\mathbf{m}}) \| \mathbf{v}_2) - \mathbf{u}\mathbf{y} \\
&= (\mathbf{v}_1 + \tilde{\mathbf{e}} \| \mathbf{v}_2) - \mathbf{u}\mathbf{y} \\
&= \mathbf{v} + \tilde{\mathbf{e}} - \mathbf{u}\mathbf{y} \\
&= \mathbf{m}\mathbf{G}_1 + \mathbf{s}\mathbf{r}_2 + \mathbf{e} + \tilde{\mathbf{e}} - (\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2)\mathbf{y} \\
&= \mathbf{m}\mathbf{G}_1 + \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e} + \tilde{\mathbf{e}}.
\end{aligned}$$

Since $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{e}, \tilde{\mathbf{e}}$ all have low Hamming weight, the equivalent error in $\mathcal{C}_1$ is $\mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e} + \tilde{\mathbf{e}}$ with relatively low weight. Decryption succeeds if this equivalent error is within the error-correcting capability of $\mathcal{C}_1$.

5.2 HARE-KEM

As in HQC, HARE-KEM is constructed from HARE-PKE using the FO transformation [15] and the HHK framework with implicit rejection [17]. The formal KEM procedures are provided in Appendix C.3.

Instance	NIST Category	Target Security (bits)	n	k	$w = w_r$	w_e	n_1	n_2	α	DFR
HARE-1	NIST-1	128	16067	128	66	105	41	384	7	$< 2^{-128}$
HARE-3	NIST-3	192	32653	192	100	163	51	640	11	$< 2^{-194}$
HARE-5	NIST-5	256	52379	256	131	224	81	640	9	$< 2^{-258}$

Table 3: The parameter choices for HARE.

5.3 Parameter Choices

The HARE parameter sets are summarized in Table 3, which includes the target security level, the quasi-cyclic dimension n, and the message dimension k. The parameters w, w_r and w_e represent the Hamming weight of vectors $(\mathbf{x}, \mathbf{y})$, $(\mathbf{r}_1, \mathbf{r}_2)$ and $\mathbf{e}$, respectively. For code choices, n_1 denotes the length of RS code, n_2 denotes the length of duplicated RM code and α denotes the threshold in the erasure decoding procedure. We upper bound the DFR for each parameter set.

The resulting sizes and bandwidth are summarized in Table 4. While satisfying the same security requirements as HQC, HARE achieves smaller overall sizes across all three security levels, reducing the bandwidth by 15.4%, 14.8%, and 15.3%, respectively.

Instance	Target Security (bits)	Encapsulation Key	Decapsulation Key	Ciphertext	Bandwidth
HARE-1	128	2041	2137	3608	5649
HQC-1		2241	2321	4433	6674
HARE-3	192	4114	4210	7378	11492
HQC-3		4514	4602	8978	13492
HARE-5	256	6580	6676	11774	18354
HQC-5		7237	7333	14421	21658

Table 4: Encapsulation-key, decapsulation-key, and ciphertext sizes, and total bandwidth, for HARE-KEM and HQC-KEM. All values are expressed in bytes.

6 Security Proof

6.1 Security Impact of HARE's Design Changes

Compared with the original HQC scheme, HARE introduces two modifications that require separate treatment in the security analysis. We discuss their effects before presenting the formal security results.

Unbalanced Weights. First, HARE adopts unbalanced Hamming weights for $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}$, which provides the adversary with additional information about the

error distribution. As shown in [38], this prior knowledge can be exploited by Information Set Decoding (ISD) with hints, potentially reducing the complexity of solving the corresponding SD problem. To quantify the resulting security loss, we formulate the SD problem with an unbalanced error distribution and relate it to standard SD in Section 6.2. Although the practical speedup of ISD with hints may be smaller than the theoretical bound, we conservatively deduct the full loss given by the reduction. The resulting concrete estimates for 3-DUQCSD-PT are presented in Section 6.5.

Ciphertext Compression. Second, HARE introduces ciphertext compression based on covering codes, which modifies the ciphertext representation during encapsulation and decapsulation. Because compression depends only on public parameters, any party can apply it without secret information. It can therefore be treated as a publicly computable transformation that introduces no additional security assumption.

6.2 Unbalanced Syndrome Decoding Problem

We define the standard syndrome decoding problem (SD) and its unbalanced variant (USD), and then give a reduction from SD to USD.

Definition 11 (s-SD Problem). *Let n, s, t be positive integers. Sample $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(sn-n)\times sn}$ and $\mathbf{e} \xleftarrow{\$} \mathbb{F}_2^{sn}$ subject to $w_H(\mathbf{e}) = t$, and compute $\mathbf{s}^\top = \mathbf{H}\mathbf{e}^\top$. Given $(\mathbf{H}, \mathbf{s})$, the goal of the $\mathsf{s\text{-}SD}(n,t)$ problem is to find $\mathbf{e}' \in \mathbb{F}_2^{sn}$ satisfying $w_H(\mathbf{e}') = t$ and $\mathbf{H}\mathbf{e}'^\top = \mathbf{s}^\top$.*

Definition 12 (USD Problem). *Let n, w_0, w_1 be positive integers. Sample $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{2n\times 3n}$ and $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \xleftarrow{\$} \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n$ subject to $w_H(\mathbf{e}_0) = w_H(\mathbf{e}_2) = w_0$ and $w_H(\mathbf{e}_1) = w_1$, and compute $\mathbf{s}^\top = \mathbf{H}(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^\top$. Given $(\mathbf{H}, \mathbf{s})$, the goal of the $\mathsf{USD}(n, w_0, w_1)$ problem is to find $(\mathbf{e}'_0, \mathbf{e}'_1, \mathbf{e}'_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n$ satisfying $w_H(\mathbf{e}'_0) = w_H(\mathbf{e}'_2) = w_0$, $w_H(\mathbf{e}'_1) = w_1$, and*

$$\mathbf{H}(\mathbf{e}'_0, \mathbf{e}'_1, \mathbf{e}'_2)^\top = \mathbf{s}^\top.$$

Theorem 4. *Let $t = 2w_0 + w_1$. If an adversary solves $\mathsf{USD}(n, w_0, w_1)$ with success probability ε, then there exists an adversary solving 3-$\mathsf{SD}(n,t)$ with success probability at least*

$$\varepsilon' = \frac{\binom{n}{w_0}^2 \binom{n}{w_1}}{\binom{3n}{t}} \varepsilon.$$

Proof. Let $\mathcal{A}$ solve $\mathsf{USD}(n, w_0, w_1)$ with success probability ε. We construct $\mathcal{A}'$ for 3-$\mathsf{SD}(n,t)$. Given $(\mathbf{H}, \mathbf{s})$, $\mathcal{A}'$ samples a uniformly random permutation matrix $\mathbf{P} \in \mathbb{F}_2^{3n\times 3n}$ and invokes $\mathcal{A}$ on $(\mathbf{HP}, \mathbf{s})$. If $\mathcal{A}$ returns $\mathbf{e}' = (\mathbf{e}'_0, \mathbf{e}'_1, \mathbf{e}'_2)$ satisfying the USD weight constraints and $(\mathbf{HP})\mathbf{e}'^\top = \mathbf{s}^\top$, then $\mathcal{A}'$ returns $\mathbf{e}'\mathbf{P}^\top$. Because $\mathbf{H}$ is uniform, $\mathbf{HP}$ has the same distribution as $\mathbf{H}$. A uniformly random permutation

places exactly w_0, w_1, w_0 nonzero entries of a uniform weight-t vector into the three length-n blocks with probability

$$\Pr[\mathsf{B}] = \frac{\binom{n}{w_0}^2 \binom{n}{w_1}}{\binom{3n}{t}}.$$

Conditioned on B, $(\mathbf{HP}, \mathbf{s})$ follows the USD distribution. Moreover,

$$\mathbf{H}(\mathbf{e}'\mathbf{P}^\top)^\top = \mathbf{HPe}'^\top = \mathbf{s}^\top,$$

and the returned vector has weight t. Hence, $\mathcal{A}'$ succeeds with probability at least $\Pr[\mathsf{B}]\varepsilon = \varepsilon'$. □

6.3 Security Assumption

The security of our scheme relies on the 2-DQCSD-P problem and 3-DUQCSD-PT problem. The 2-DQCSD-P problem is identical to the corresponding hardness assumption in HQC. The 3-DUQCSD-PT problem is formulated to characterize the additional distribution information introduced by the unbalanced error weights in HARE.

For $b_1 \in \{0,1\}$, let $\mathbb{F}_{2,b_1}^n = \{\mathbf{h} \in \mathbb{F}_2^n \mid w_H(\mathbf{h}) \equiv b_1 \pmod 2\}$, i.e., the set of all length-n binary vectors with parity b_1. Let

$$\mathbb{F}_{2,b_1}^{n\times 2n} = \{\mathbf{H} = (\mathbf{I}_n, \mathtt{rot}(\mathbf{h})) \in \mathbb{F}_2^{n\times 2n} | \mathbf{h} \in \mathbb{F}_{2,b_1}^n\}, \text{and}$$

$$\mathbb{F}_{2,b_1,b_2}^{2n\times 3n} = \left\{ \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathtt{rot}(\mathbf{h}_1) \\ \mathbf{0} & \mathbf{I}_n & \mathtt{rot}(\mathbf{h}_2) \end{pmatrix} \in \mathbb{F}_2^{2n\times 3n} \;\middle|\; \mathbf{h}_1 \in \mathbb{F}_{2,b_1}^n,\ \mathbf{h}_2 \in \mathbb{F}_{2,b_2}^n \right\}.$$

Definition 13 (2-QCSD-P Distribution). *Let n, w be positive integers and let $b_1 \in \{0,1\}$. Define $b_2 = (w + b_1 w) \bmod 2$. The 2-Quasi-Cyclic Syndrome Decoding with Parity Distribution* $\mathsf{2\text{-}QCSD\text{-}P}(n, w, b_1, b_2)$ *samples $\mathbf{H} \xleftarrow{\$} \mathbb{F}_{2,b_1}^{n\times 2n}$ and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \xleftarrow{\$} \mathbb{F}_2^{2n}$ such that $w_H(\mathbf{e}_1) = w_H(\mathbf{e}_2) = w$, computes $\mathbf{s}^\top = \mathbf{He}^\top$ and outputs $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_{2,b_1}^{n\times 2n} \times \mathbb{F}_{2,b_2}^n$.*

Definition 14 (2-DQCSD-P Problem). *Let n, w be positive integers and let $b_1 \in \{0,1\}$. Define $b_2 = (w + b_1 w) \bmod 2$. Given $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_{2,b_1}^{n\times 2n} \times \mathbb{F}_{2,b_2}^n$, the Decisional 2-Quasi-Cyclic Syndrome Decoding with Parity Problem* $\mathsf{2\text{-}DQCSD\text{-}P}$ (n, w, b_1, b_2) *asks to decide with non-negligible advantage whether the pair $(\mathbf{H}, \mathbf{s})$ came from the* $\mathsf{2\text{-}QCSD\text{-}P}(n, w, b_1, b_2)$ *distribution or uniform distribution over $\mathbb{F}_{2,b_1}^{n\times 2n} \times \mathbb{F}_{2,b_2}^n$.*

Definition 15 (3-UQCSD-PT Distribution). *Let n, w_0, w_1, l be positive integers with $l < n$, and let $b_1, b_2 \in \{0,1\}$. Define $b_3 = (w_0 + b_1 w_0) \bmod 2$. The 3-Unbalanced-Quasi-Cyclic Syndrome Decoding with Parity and Truncation distribution is defined as follows.* $\mathsf{3\text{-}UQCSD\text{-}PT}(n, w_0, w_1, b_1, b_2, b_3, l)$ *samples $\mathbf{H} \xleftarrow{\$} \mathbb{F}_{2,b_1,b_2}^{2n\times 3n}$ and $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \xleftarrow{\$} \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n$ such that $w_H(\mathbf{e}_0) = w_H(\mathbf{e}_2) = w_0$ and $w_H(\mathbf{e}_1) = w_1$, computes $\mathbf{s}^T = \mathbf{He}^T$ where $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1)$ and outputs $(\mathbf{H}, (\mathbf{s}_0, \mathsf{Truncate}(\mathbf{s}_1, l))) \in \mathbb{F}_{2,b_1,b_2}^{2n\times 3n} \times (\mathbb{F}_{2,b_3}^n \times \mathbb{F}_2^l)$.*

Definition 16 (3-DUQCSD-PT Problem). *Let n, w_0, w_1, l be positive integers with $l \leq n$, and let $b_1, b_2 \in \{0,1\}$. Define $b_3 = (w_0 + b_1 w_0) \bmod 2$. Given $(\mathbf{H}, (\mathbf{s}_0, \mathbf{s}_1)) \in \mathbb{F}_{2,b_1,b_2}^{2n\times 3n} \times (\mathbb{F}_{2,b_3}^{n} \times \mathbb{F}_2^{l})$, the Decisional 3-Unbalanced-Quasi-Cyclic Syndrome Decoding with Parity and Truncation Problem* 3-DUQCSD-PT $(n, w_0, w_1, b_1, b_2, b_3, l)$ *asks to decide with non-negligible advantage whether $(\mathbf{H}, (\mathbf{s}_0, \mathbf{s}_1))$ comes from the* 3-UQCSD-PT $(n, w_0, w_1, b_1, b_2, b_3, l)$ *distribution or the uniform distribution over $\mathbb{F}_{2,b_1,b_2}^{2n\times 3n} \times (\mathbb{F}_{2,b_3}^{n} \times \mathbb{F}_2^{l})$.*

6.4 IND-CPA and IND-CCA2 Security

The IND-CPA security of HARE-PKE can be reduced to the hardness of the corresponding SD problems in Theorem 5. The detailed proof is provided in Appendix D.

Theorem 5. *For any* IND-CPA *adversary $\mathcal{A}$ against the* HARE-PKE *scheme, there exist adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HARE\text{-}PKE}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{2\text{-}DQCSD\text{-}P}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{3\text{-}DUQCSD\text{-}PT}}(\mathcal{B}_2). \tag{4}$$

Following the same salted Fujisaki-Okamoto transformation as in HQC, the IND-CCA2 security of HARE-KEM follows from the IND-CPA security of HARE-PKE and the correctness of the underlying PKE. The resulting bound is stated in Theorem 6.

Theorem 6. *If* HARE-PKE *is δ-correct, then for any* IND-CCA2 *adversary $\mathcal{A}$ against the* HARE-KEM *scheme issuing at most q_{RO} queries to* G *and q_D queries to the* HARE-KEM.Decaps *oracle, there exist adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{HARE\text{-}KEM}}(\mathcal{A}) \leq & \frac{1}{2^{|\mathbf{K}|}\cdot 2^{|\mathbf{salt}|}} + \frac{3q_{RO}}{2^{|\mathbf{K}|}} + (q_{RO}+q_D)\cdot\delta \\ & + 2\cdot\left(\mathsf{Adv}_{\mathsf{2\text{-}DQCSD\text{-}P}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{3\text{-}DUQCSD\text{-}PT}}(\mathcal{B}_2)\right).\end{aligned} \tag{5}$$

6.5 Known Attacks

We divide the potential attacks into two main categories. First, **structural attacks** exploit the cyclic properties of quasi-cyclic codes when $X^n - 1$ possesses low-degree factors [20,23]. To mitigate these threats, we select n as a primitive prime, ensuring that the polynomial $X^n - 1$ has only two irreducible factors. Consequently, such structural attacks cannot be efficiently applied to our scheme.

Second, we consider attacks targeting the 2-DQCSD-P and 3-DUQCSD-PT problems. The most efficient methods for solving the SD problem are ISD algorithms, such as those presented in [31,35,25,6]. In the quasi-cyclic setting, we also account for the Decoding One Out of Many (DOOM) speedup [32], which provides an $O(\sqrt{n})$ complexity gain for both 2-DQCSD-P and 3-DUQCSD-PT. For 3-DUQCSD-PT, the corresponding search problem is the unbalanced syndrome decoding problem $\mathsf{USD}(n, w_0, w_1)$, whose three error blocks have weights (w_0, w_1, w_0). Instead of estimating this USD problem directly, we first use the

open-source estimator from [13] to compute the ISD security of its corresponding balanced SD problem, where the total weight is $2w_0 + w_1$. The considered ISD algorithms include Prange [31], Stern [7,29,35], BJMM [6,14,25], Both-May [11] and May-Ozerov [26].

This ISD estimate already accounts for the $O(\sqrt{n})$ quasi-cyclic DOOM speedup. We then translate the resulting SD security into an estimate for USD using the reduction in Section 6.2. More precisely, the reduction introduces a security loss of

$$\log_2 \left(\frac{\binom{3n}{2w_0+w_1}}{\binom{n}{w_0}^2 \binom{n}{w_1}} \right)$$

bits, which we subtract from the estimated security of the corresponding SD problem. Since this calculation assumes that the adversary obtains the full theoretical advantage allowed by the reduction, the resulting USD security is a conservative estimate. We use this value as the concrete security estimate for 3-DUQCSD-PT in Table 5.

Instance	Problem	Prange	Stern	BJMM	Both-May	May-Ozerov	Best	Target Security
HARE-1	3-DUQCSD-PT	158	137	137	137	137	**137**	128
	2-DQCSD-P	166	145	145	145	145	**145**	
HARE-3	3-DUQCSD-PT	227	204	204	204	204	**204**	192
	2-DQCSD-P	237	213	213	213	213	**213**	
HARE-5	3-DUQCSD-PT	291	266	266	265	265	**265**	256
	2-DQCSD-P	300	275	275	275	275	**275**	

Table 5: The ISD bit-security of HARE parameter sets.

7 Decryption Failure Rate

Following the correlated-hypergeometric model of Bitzer et al. [8], we model the coefficients of each cyclic product of weight-a and weight-b vectors as hypergeometric random variables conditioned on their sum being ab, and then reduce them modulo 2. This model accounts for the dependence between the coefficients that is omitted by the independent Bernoulli approximation used in [16].

As in [8, Lemma 2], the resulting distributions are radially symmetric. Hence, conditioned on Hamming weight w, a vector is uniformly distributed over the Hamming sphere

$$\mathcal{S}_w^n := \{\mathbf{z} \in \mathbb{F}_2^n \mid w_H(\mathbf{z}) = w\}.$$

We use this property throughout the following analysis.

7.1 Error Distribution Model

Recall that, before truncation, the equivalent error in HARE is

$$\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e} + \tilde{\mathbf{e}}.$$

Let $w_0 = w_H(\mathbf{x}) = w_H(\mathbf{y})$, $w_1 = w_H(\mathbf{r}_1) = w_H(\mathbf{r}_2)$, $w_2 = w_H(\mathbf{e})$. For the compression contribution, let $w_3 = BR_{\mathrm{cc}}$ be the maximum Hamming weight introduced by the B KR blocks. We represent this contribution by an independent vector uniformly distributed over $\mathcal{S}^n_{w_3}$ and incorporate it through the same fixed-weight convolution.

For $a, b, t \in \{0, \ldots, n\}$, define $p_{a,b,t} = \Pr[w_H(\mathbf{u} \cdot \mathbf{v}) = t]$, where $\mathbf{u} \xleftarrow{\$} \mathcal{R}_a$ and $\mathbf{v} \xleftarrow{\$} \mathcal{R}_b$ are sampled independently. The values $p_{a,b,t}$ can be computed using the method of [8]. The weight distribution of the equivalent error is derived in Theorem 7, and the effect of truncation is incorporated in Theorem 8.

Theorem 7. *Assume that all vectors lie in $\mathbb{F}_2^n$. We have*

$$\Pr[w_H(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1) = w] = \sum_{l,s} \frac{\binom{s}{l}\binom{n-s}{w+l-s}}{\binom{n}{w+2l-s}} p_{w_0,w_1,s} \cdot p_{w_0,w_1,w+2l-s}.$$

Moreover,

$$\Pr[w_H(\mathbf{x}\cdot\mathbf{r}_2 - \mathbf{y}\cdot\mathbf{r}_1 + \mathbf{e}) = w] = \sum_{l} \frac{\binom{w_2}{l}\binom{n-w_2}{w+l-w_2}}{\binom{n}{w+2l-w_2}} \Pr[w_H(\mathbf{x}\cdot\mathbf{r}_2 - \mathbf{y}\cdot\mathbf{r}_1) = w+2l-w_2].$$

Finally, after incorporating the modeled weight-w_3 compression error, we obtain

$$\Pr[w_H(\mathbf{e}') = w] = \sum_{l} \frac{\binom{w_3}{l}\binom{n-w_3}{w+l-w_3}}{\binom{n}{w+2l-w_3}} \Pr[w_H(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}) = w + 2l - w_3].$$

Proof. The first identity follows from the weight distribution of the sum of two independent vectors under the hypergeometric model. For vectors of weights s and $t = w + 2l - s$, their sum has weight w exactly when their supports overlap in l positions. The second and third identities follow from the same argument by successively adding vectors of fixed weights w_2 and w_3. □

Theorem 8. *Taking truncation into account, for every $w \in \{0, \ldots, l_1\}$,*

$$\Pr[w_H(\mathsf{Truncate}(\mathbf{e}', l_1)) = w] = \sum_{l=w}^{w+n-l_1} \frac{\binom{l_1}{w}\binom{n-l_1}{l-w}}{\binom{n}{l}} \Pr[w_H(\mathbf{e}') = l].$$

Proof. Condition on $w_H(\mathbf{e}') = l$. The first l_1 coordinates contain exactly w nonzero entries in $\binom{l_1}{w}\binom{n-l_1}{l-w}$ weight-l patterns, out of $\binom{n}{l}$ such patterns in total. Summing over all admissible values of l gives the result. □

7.2 DFR of RM-RS Codes

Let the outer RS code have length $n_e := n_1$ and let each duplicated RM block have length $n_i := n_2$. The binary length of the concatenated code is therefore $l_1 = n_e n_i$. For a total error weight w on these l_1 coordinates, let Γ_w be the conservative failure count defined in Theorem 2.

Theorem 9. *Under the hypergeometric model above, the DFR of HARE is upper bounded by*

$$\mathsf{DFR} \le \sum_{w=0}^{l_1} \Pr[w_H(\mathsf{Truncate}(\mathbf{e}', l_1)) = w] \, \frac{\min\left\{\Gamma_w, \binom{l_1}{w}\right\}}{\binom{l_1}{w}}.$$

Proof. Conditioned on the truncated error having weight w, the number of patterns conservatively classified as failures is at most Γ_w and cannot exceed $\binom{l_1}{w}$. Thus, the conditional failure probability is upper bounded by

$$\frac{\min\left\{\Gamma_w, \binom{l_1}{w}\right\}}{\binom{l_1}{w}}.$$

Averaging this bound over the truncated weight distribution from Theorem 8 proves the result. □

8 Implementation

8.1 Experimental Setup

We benchmark the HARE and HQC v5.0.0 implementations at security categories 1, 3, and 5. The experiments were conducted on an Alibaba Cloud `ecs.hfc8i.xlarge` instance with an Intel Xeon Gold 6462C processor under KVM virtualization, running Ubuntu 22.04 with Linux kernel 5.15. The guest exposed two logical CPUs, and all measurements were pinned to logical CPU 1. The instance used shared host resources, and the guest did not control the CPU frequency governor or turbo settings.

Both schemes were compiled with GCC 11.4.0 using the same flags for each implementation class. For each operation, we collected 1,000 samples over five sessions, each containing eight blocks of twenty-five samples. The twelve binaries were executed in randomized order within each block, with eight warmup iterations per invocation. We report the median of the 1,000 samples for each operation in each binary. All samples completed successfully without CPU migration. All timings cover complete KEM operations, including SHA/XOF computations. Decapsulation uses the default serialized secret-key format without a resident secret context or pre-imported state. Execution costs were measured using the invariant time-stamp counter (TSC) and are reported in thousands of TSC ticks, not dynamic core cycles.

8.2 Reference Implementation

The performance of the reference implementations on the aforementioned benchmark platform is given in Table 6. Both implementations are written in C and were compiled using the flags `-DNDEBUG -O3 -std=c99`.

Instance	KeyGen	Encaps	Decaps
HARE-1	3 673.733	7 789.907	12 013.527
HQC-1	3 603.277	7 208.016	11 022.354
HARE-3	11 141.260	23 152.381	35 028.659
HQC-3	10 950.043	21 889.549	33 118.013
HARE-5	22 253.258	45 997.474	70 046.756
HQC-5	26 484.111	52 960.318	80 018.191

Table 6: Performances (in thousands of TSC ticks) of the reference implementations of HARE and HQC.

At security categories 1 and 3, HARE has execution costs close to those of HQC, requiring approximately 1.7% to 9.0% more ticks across the three operations. At security category 5, HARE reduces the costs of key generation, encapsulation, and decapsulation by approximately 16.0%, 13.1%, and 12.5%, respectively. The reference trend reflects size-dependent Karatsuba leaf costs, with additional KR compression and erasure processing in Encaps and Decaps. It is consistent with HARE-5 using smaller leaves at the same recursion depth, whereas parameter reduction does not yield a uniform speedup at categories 1 and 3.

8.3 Optimized Implementation

The performance of the optimized implementations on the benchmark platform is given in Table 7. Both implementations use AVX2 instructions and were compiled using the flags `-DNDEBUG -O3 -std=c99 -flto -funroll-all-loops` and `-mavx -mavx2 -mbmi -mpclmul`. The optimized HARE implementation accelerates polynomial multiplication, concatenated-code processing, and KR compression using vectorized arithmetic and specialized routines. These optimizations preserve the parameters and serialized interface.

The current optimized HARE implementation is slower than HQC in all reported operations except key generation at category 1. Encapsulation and decapsulation require approximately 19-29% and 31-44% more TSC ticks, respectively. By comparison, the reference implementations differ by at most 9.0% at categories 1 and 3, while HARE is faster at category 5. These reference results show that HARE incorporates compression and erasure-aware decoding with limited overall runtime overhead relative to HQC at the selected parameter sets. The larger performance gap in the optimized implementations suggests room for further implementation improvements. Performance optimization is ongoing, with vectorized erasure-aware decoding, parameter-specific scheduling of binary polynomial multiplication, and the KR compression path for blocks remaining after full batches as candidates for further improvement.

Instance	KeyGen	Encaps	Decaps
HARE-1	61.999	150.225	383.038
HQC-1	63.400	126.181	292.048
HARE-3	169.317	381.170	827.840
HQC-3	151.163	301.403	617.541
HARE-5	351.980	784.097	1 738.938
HQC-5	304.706	608.859	1 206.089

Table 7: Performances (in thousands of TSC ticks) of the optimized AVX2 implementations of HARE and HQC.

9 Conclusion and Future Work

In this work, we present HARE, which combines erasure decoding with covering-code-based compression to reduce the sizes of HQC. Erasure decoding exploits reliability information to provide additional tolerance to compression-induced errors, while covering codes enable a more compact representation of the transmitted data. By jointly designing these two components, HARE reduces the combined public-key and ciphertext size by approximately 15% compared with HQC, while providing an IND-CCA2 security reduction and meeting the target concrete-security and DFR bounds. Our results indicate that co-designing decoding and compression can improve the compactness of HQC-style code-based KEMs.

Several directions remain for future work. First, the decoder could be improved through adaptive threshold selection or richer reliability-aware erasure rules. Second, more flexible compression schemes could be constructed using non-uniform combinations of different covering codes, potentially outperforming the current construction based on direct products of KR codes. Finally, the DFR analysis could be refined by accounting for the potentially non-uniform noise introduced by covering-code compression, leading to tighter estimates and better parameter selection.

References

1. National institute of standards and technology: Post-quantum cryptography project. https://csrc.nist.gov/projects/post-quantum-cryptography (2016)
2. Aguilar-Melchor, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. IEEE Transactions on Information Theory **64**(5), 3927–3943 (2018)
3. Annechini, A., Barenghi, A., Pelosi, G.: On the decoding failure rate of hqc. Cryptology ePrint Archive (2026)
4. Aragon, N., Barreto, P.S., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: bit flipping key encapsulation (2017)
5. Baldi, M., Bitzer, S., Lilla, N., Santini, P.: Hqc beyond the bsc: Towards error structure-aware decoding. In: Code-Based Cryptography Workshop. pp. 1–24. Springer (2024)
6. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How 1+1=0 improves information set decoding. In: Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. pp. 520–536. Springer (2012)
7. Bernstein, D.J., Lange, T., Peters, C.: Attacking and defending the McEliece cryptosystem. In: Buchmann, J., Ding, J. (eds.) Post-quantum cryptography, second international workshop, PQCRYPTO 2008. pp. 31–46. Springer, Heidelberg, Germany, Cincinnati, Ohio, United States (Oct 17–19, 2008). https://doi.org/10.1007/978-3-540-88403-3_3
8. Bitzer, S., Deneuville, J.C., Munisamy, E., Purtipli, B., Ritterhoff, S., Wachter-Zeh, A.: Hqc beyond the standard: Ciphertext compression and refined dfr analysis. Cryptology ePrint Archive (2025)
9. Bitzer, S., Purtipli, B., Wachter-Zeh, A.: Improved codes and decoders for hqc. Cryptology ePrint Archive (2026)
10. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European symposium on security and privacy (EuroS&P). pp. 353–367. IEEE (2018)
11. Both, L., May, A.: Decoding linear codes with high error rate and its impact for LPN security. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018. pp. 25–46. Springer, Heidelberg, Germany, Fort Lauderdale, Florida, United States (Apr 9–11, 2018). https://doi.org/10.1007/978-3-319-79063-3_2
12. Davydov, A.A., Marcugini, S., Pambianco, F.: New upper bounds for binary linear covering codes. arXiv preprint arXiv:2511.02542 (2025)
13. Esser, A., Verbel, J.A., Zweydinger, F., Bellini, E.: {SoK: CryptographicEstimators - a Software Library for Cryptographic Hardness Estimation}. In: AsiaCCS. {ACM} (2024)
14. Esser, A., Zweydinger, F.: New time-memory trade-offs for subset sum – improving ISD in theory and practice. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part V. pp. 360–390. Springer, Heidelberg, Germany (2023)
15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual international cryptology conference. pp. 537–554. Springer (1999)

16. Gaborit, P., Aguilar-Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Persichetti, E., Zémor, G., Bos, J., et al.: Hamming quasi-cyclic (hqc). NIST Post-Quantum Standardization, 4th Round (2025)
17. Glabush, L., Hövelmanns, K., Stebila, D.: Tight multi-challenge security reductions for key encapsulation mechanisms. IACR Cryptol. ePrint Arch. **2025**, 343 (2025)
18. Guan, C., Luo, L., Jiang, H., Hou, J., Yu, T., Wang, H., Li, K., Qu, L.: Compact hqc with new (un) balance. Cryptology ePrint Archive (2026)
19. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 1–20. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014). https://doi.org/10.1007/978-3-662-45611-8_1
20. Guo, Q., Johansson, T., Löndahl, C.: A new algorithm for solving ring-lpn with a reducible polynomial. IEEE Transactions on Information Theory **61**(11), 6204–6212 (2015)
21. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)
22. Kaikkonen, M.K., Rosendahl, P.: New covering codes from an ads-like construction. IEEE Transactions on Information Theory **49**(7), 1809–1812 (2003)
23. Löndahl, C., Johansson, T., Koochak Shooshtari, M., Ahmadian-Attari, M., Aref, M.R.: Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension. Designs, Codes and Cryptography **80**(2), 359–377 (2016)
24. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1977)
25. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in o(2$\hat{0}$.054n). In: Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17. pp. 107–124. Springer (2011)
26. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 203–228. Springer (2015)
27. McEliece, R.J.: A public-key cryptosystem based on algebraic. Coding Thv **4244**, 114–116 (1978)
28. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 333–342 (2009)
29. Peters, C.: Information-set decoding for linear codes over F_q. In: Sendrier, N. (ed.) The Third International Workshop on Post-Quantum Cryptography, PQCRYPTO 2010. pp. 81–94. Springer, Heidelberg, Germany, Darmstadt, Germany (May 25–28, 2010). https://doi.org/10.1007/978-3-642-12929-2_7
30. Pöppelmann, T., Güneysu, T.: Towards practical lattice-based public-key encryption on reconfigurable hardware. In: International Conference on Selected Areas in Cryptography. pp. 68–85. Springer (2013)
31. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **8**(5), 5–9 (1962)
32. Sendrier, N.: Decoding one out of many. In: International Workshop on Post-Quantum Cryptography. pp. 51–67. Springer (2011)
33. Sendrier, N., Vasseur, V.: On the decoding failure rate of qc-mdpc bit-flipping decoders. In: International Conference on Post-Quantum Cryptography. pp. 404–416. Springer (2019)

34. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society Press, Santa Fe, NM, USA (Nov 20–22, 1994). https://doi.org/10.1109/SFCS.1994.365700
35. Stern, J.: A method for finding codewords of small weight. In: Coding Theory and Applications: 3rd International Colloquium Toulon, France, November 2–4, 1988 Proceedings 3. pp. 106–113. Springer (1989)
36. Tillich, J.P.: The decoding failure probability of mdpc codes. In: 2018 IEEE International Symposium on Information Theory (ISIT). pp. 941–945. IEEE (2018)
37. Vasseur, V.: Qc-mdpc codes dfr and the ind-cca security of bike. HAL (2022)
38. Wang, T., Wang, A., Wang, X.: Exploring decryption failures of bike: new class of weak keys and key recovery attacks. In: Annual International Cryptology Conference. pp. 70–100. Springer (2023)

Appendix

A Experiments for Erasure Decoding of RM Codes

We evaluate the conservative estimates from Theorem 1 for the duplicated RM$[384, 8, 192]$ code and decoding threshold $\alpha = 7$ used in HARE-1. For each tested error weight w, we independently sample $100,000$ weight-w error patterns uniformly at random, apply Decode' to each pattern, and record whether the decoder returns an incorrect codeword, an erasure, or the correct codeword. For each outcome type, we report all non-trivial error weights, namely those for which the corresponding event occurs in at least one but not all of the $100,000$ trials. We compare the resulting empirical frequencies with the normalized theoretical quantities

$$\frac{\mathrm{N}_{\mathsf{error},w}}{\binom{n}{w}} \quad \text{and} \quad \frac{\mathrm{N}_{\mathsf{error},w} + \mathrm{N}_{\mathsf{erasure},w}}{\binom{n}{w}},$$

which correspond to the estimated error and non-correct rates, respectively.

Figure 2 compares the theoretical estimate of the error rate with the empirical rate observed in simulation. Across the tested range of w, the theoretical curve consistently remains above the empirical curve. Similarly, Figure 3 compares the theoretical and empirical rates of non-correct outputs, including both errors and erasures, and exhibits the same behavior. Although the theoretical estimates are relatively conservative, replacing $\mathrm{N}_{\mathsf{error},w}$, $\mathrm{N}_{\mathsf{erasure},w}$, and $\mathrm{N}_{\mathsf{correct},w}$ with their empirically measured values improves the resulting DFR estimate by only about 2 bits, which has little impact on the selection of the security parameters. We therefore retain the more conservative theoretical estimates in our parameter analysis. Overall, the estimates in Theorem 1 provide conservative upper bounds on both the number of errors and the number of non-correct outputs.

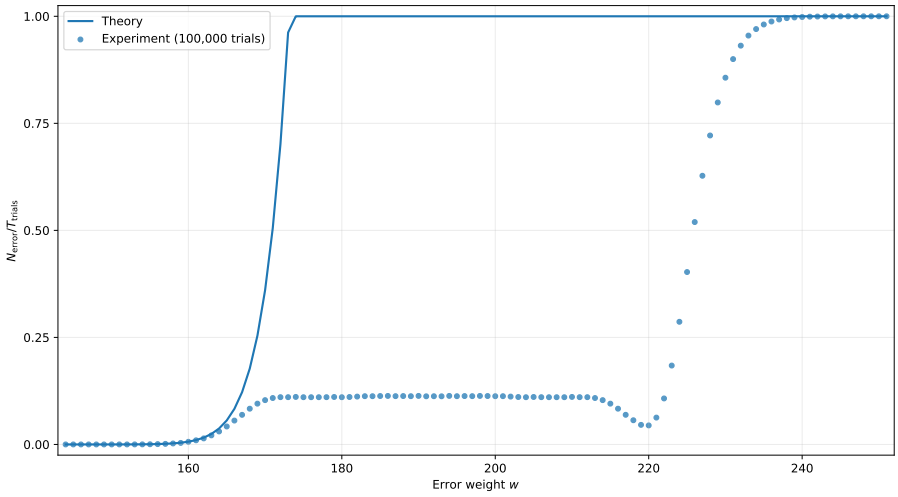

Fig. 2: Theoretical and empirical error rates of Decode' for HARE-1.

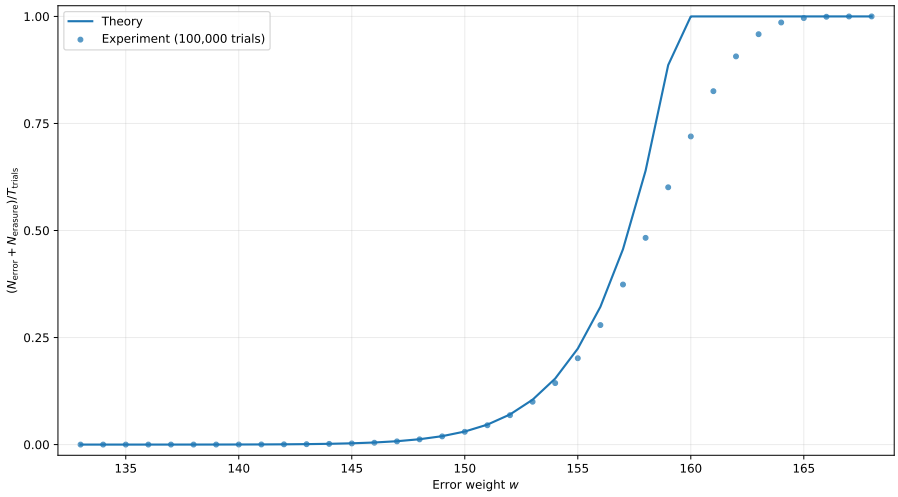

Fig. 3: Theoretical and empirical non-correct rates of Decode' for HARE-1. Non-correct outcomes include both errors and erasures.

B Reed-Solomon Codes with Erasure Decoding

Let $\rho = n - k$. We obtain the RS$[n, k, \rho + 1]$ codes used in HARE by shortening systematic information coordinates of a full-length primitive narrow-sense Reed-Solomon code over $\mathbb{F}_q$. Shortening reduces the length and dimension by the same amount, so the resulting code remains maximum distance separable (MDS) and has minimum distance $d_{\min} = \rho + 1$. It therefore corrects every pattern of g errors and h known-location erasures satisfying $2g + h \leq \rho$. The parameters used by the three HARE instances are listed in Table 8.

Constructing and Encoding Shortened Reed-Solomon Codes. HARE uses

$$\mathbb{F}_{2^8} = \mathbb{F}_2[Y]/(P(Y)), \qquad P(Y) = Y^8 + Y^4 + Y^3 + Y^2 + 1,$$

where P is primitive, and $\zeta = [Y]$ is therefore a primitive element of $\mathbb{F}_{2^8}$. The generator polynomial is

$$G_{\mathrm{RS}}(x) = \prod_{i=1}^{\rho} (x + \zeta^i).$$

The shortened code uses the same systematic remainder rule as the full-length code. Representing the message by $u(x) = u_0 + \cdots + u_{k-1}x^{k-1}$, let $b(x)$ be the remainder of $x^\rho u(x)$ upon division by $G_{\mathrm{RS}}(x)$. Then

$$c(x) = b(x) + x^\rho u(x),$$

so the first ρ coefficients are parity symbols and the following k coefficients are the message.

Decoding Shortened Reed-Solomon Codes. Let $\mathcal{J}, \mathcal{L} \subseteq \{0, \ldots, n-1\}$ be the sets of unknown error and known erasure positions, respectively, with $|\mathcal{J}| = g$, $|\mathcal{L}| = h$, and $\mathcal{J} \cap \mathcal{L} = \varnothing$. Before computing syndromes, each erased symbol is replaced by a fixed field element. For every $p \in \mathcal{J} \cup \mathcal{L}$, let ε_p be the resulting discrepancy from the transmitted symbol and let $X_p = \zeta^p$ be its location number. Thus $\varepsilon_p \neq 0$ for $p \in \mathcal{J}$, while it may be zero for $p \in \mathcal{L}$. If $v(x)$ is the transmitted codeword and $r(x)$ is the received polynomial after filling erasures, then

$$r(x) = v(x) + \sum_{p \in \mathcal{J} \cup \mathcal{L}} \varepsilon_p x^p. \tag{6}$$

The following six steps give the decoder together with the derivation of its nontrivial operations.

Code	n	k	$d_{\min}$
RS-S1	41	16	26
RS-S3	51	24	28
RS-S5	81	32	50

Table 8: The parameter choices for shortened RS codes in HARE.

Step 1: syndromes. For $1 \le i \le \rho$, compute $S_i = r(\zeta^i)$. Since $v(\zeta^i) = 0$,

$$S_i = \sum_{p \in \mathcal{J} \cup \mathcal{L}} \varepsilon_p X_p^i. \tag{7}$$

Define the shifted syndrome polynomial

$$S^\star(z) = \sum_{i=1}^{\rho} S_i z^{i-1} = S_1 + S_2 z + \cdots + S_\rho z^{\rho-1}. \tag{8}$$

As a formal power series truncated modulo z^ρ,

$$S^\star(z) \equiv \sum_{p \in \mathcal{J} \cup \mathcal{L}} \frac{\varepsilon_p X_p}{1 + X_p z} \pmod{z^\rho}, \tag{9}$$

where each fraction denotes the truncated power-series inverse.

Step 2: modified syndromes and error locator. Form the erasure locator and modified syndrome polynomial

$$\gamma(z) = \prod_{\ell \in \mathcal{L}} (1 + X_\ell z), \tag{10}$$

$$T(z) = \big[S^\star(z)\gamma(z)\big] \bmod z^\rho = \sum_{t=0}^{\rho-1} T_t z^t. \tag{11}$$

After reduction modulo z^ρ, the contribution of each known erasure has degree at most $h-1$. Hence, for $h \le t < \rho$,

$$T_t = \sum_{p \in \mathcal{J}} \varepsilon_p X_p^{t+1} \gamma(X_p^{-1}). \tag{12}$$

Because the position sets are disjoint, $\gamma(X_p^{-1}) \neq 0$ for $p \in \mathcal{J}$. The unknown-error locator

$$\sigma(z) = \prod_{p \in \mathcal{J}} (1 + X_p z) = 1 + \sigma_1 z + \cdots + \sigma_g z^g \tag{13}$$

therefore gives the recurrence

$$T_{h+\mu} + \sigma_1 T_{h+\mu-1} + \cdots + \sigma_g T_{h+\mu-g} = 0, \qquad g \le \mu < \rho - h. \tag{14}$$

Apply Berlekamp-Massey to the $\rho - h$ consecutive terms $T_h, T_{h+1}, \ldots, T_{\rho-1}$. It recovers $\sigma(z)$ whenever $\rho - h \ge 2g$, equivalently $2g + h \le \rho$.

Step 3: unknown-error locations. Find the roots of $\sigma(z)$ and return their inverses. Since the erasure locations are already known, we scan the possible code positions directly.

Step 4: total locator and evaluator. Once all positions are known, define

$$\Lambda(z) = \sigma(z)\gamma(z), \qquad \Omega(z) = \big[S^\star(z)\Lambda(z)\big] \bmod z^\rho. \tag{15}$$

Under the decoding-radius condition, multiplication of (9) by $\Lambda(z)$ gives the ordinary polynomial identity

$$\Omega(z) = \sum_{p \in \mathcal{J} \cup \mathcal{L}} \varepsilon_p X_p \prod_{\substack{a \in \mathcal{J} \cup \mathcal{L} \\ a \neq p}} (1 + X_a z), \tag{16}$$

because its right-hand side has degree at most $g + h - 1 < \rho$.

Step 5: error and erasure values. For every $p \in \mathcal{J} \cup \mathcal{L}$, evaluation at $z = X_p^{-1}$ and differentiation of Λ give

$$\Omega(X_p^{-1}) = \varepsilon_p X_p \prod_{\substack{a \in \mathcal{J} \cup \mathcal{L} \\ a \neq p}} (1 + X_a X_p^{-1}),$$

$$\Lambda'(X_p^{-1}) = X_p \prod_{\substack{a \in \mathcal{J} \cup \mathcal{L} \\ a \neq p}} (1 + X_a X_p^{-1}).$$

The location numbers are distinct, so the common product is nonzero. Thus the error or erasure discrepancy is

$$\varepsilon_p = \frac{\Omega(X_p^{-1})}{\Lambda'(X_p^{-1})} = \frac{\Omega(\zeta^{-p})}{\Lambda'(\zeta^{-p})}. \tag{17}$$

Step 6: correction. For every located position, replace r_p by $r_p + \varepsilon_p$. If the corrected polynomial has zero syndromes at $\zeta, \ldots, \zeta^{\rho}$, return its k systematic message symbols. Otherwise, return $\perp$. Under the decoding-radius condition, this procedure restores the transmitted RS codeword.

Comparison with HQC. For comparison with the no-erasure decoding used by HQC, set

$$S_H(z) = 1 + zS^{\star}(z), \qquad Z_H(z) = \big[S_H(z)\Lambda(z)\big] \bmod z^{\rho+1}. \tag{18}$$

Within the decoding radius, $Z_H(z) = \Lambda(z) + z\Omega(z)$, and at a locator root

$$\frac{X_p Z_H(X_p^{-1})}{\Lambda'(X_p^{-1})} = \frac{\Omega(X_p^{-1})}{\Lambda'(X_p^{-1})}.$$

When $h = 0$, we have $\gamma = 1$ and $\Lambda = \sigma$. The resulting formula is the S_H/Z_H representation used by HQC.

C Formal Constructions of PKE and KEM

C.1 Auxiliary Primitives

Prior to discussing the instantiation of HARE-PKE and HARE-KEM, we introduce several auxiliary primitives used for pseudorandom byte generation, hashing, and vector sampling.

Seeds, salts, shared keys, and encryption randomness are represented as byte strings, while elements of $\mathbb{F}_2^k$ and $\mathcal{R}$ are represented as binary arrays. All lengths of byte strings, including $|\mathbf{seed}|$, $|\mathbf{salt}|$, $|\mathbf{K}|$, and $|\theta|$, are expressed in bytes.

– **Extendable Output Function:** XOF generates a pseudorandom byte string of arbitrary length from a given short seed.
– **Hash Functions:** $\mathsf{G}, \mathsf{H}, \mathsf{I}, \mathsf{J}$ denote cryptographic hash functions.
– **Vector Sampling:** $\mathsf{SampleVec}$ and $\mathsf{SampleFixedWeightVec}$ are employed to sample random vectors from the spaces $\mathcal{R}$ and $\mathcal{R}_w$, respectively.

We omit a detailed discussion of these primitives here. For concrete instantiations and implementation details, one can refer to Section 3 of [16].

C.2 PKE

We now describe the HARE-PKE scheme. The formal procedures are given in Algorithm 3, Algorithm 4, and Algorithm 5.

Algorithm 3 $\mathsf{PKE.Keygen}()$

Input: Seed $\mathbf{seed}_{\mathsf{PKE}}$
Output: Encryption key $\mathbf{ek}_{\mathsf{PKE}}$ and decryption key $\mathbf{dk}_{\mathsf{PKE}}$
1: $\mathbf{seed}_{\mathsf{PKE.dk}}, \mathbf{seed}_{\mathsf{PKE.ek}} \leftarrow \mathsf{I}(\mathbf{seed}_{\mathsf{PKE}})$
2: $\mathbf{ctx}_{\mathsf{PKE.dk}} \leftarrow \mathsf{XOF.Init}(\mathbf{seed}_{\mathsf{PKE.dk}})$
3: $\mathbf{ctx}_{\mathsf{PKE.ek}} \leftarrow \mathsf{XOF.Init}(\mathbf{seed}_{\mathsf{PKE.ek}})$
4: $\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathbf{y} \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathcal{R}_w)$
5: $\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathbf{x} \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathcal{R}_w)$
6: $\mathbf{dk}_{\mathsf{PKE}} \leftarrow \mathbf{seed}_{\mathsf{PKE.dk}}$
7: $\mathbf{ctx}_{\mathsf{PKE.ek}}, \mathbf{h} \leftarrow \mathsf{SampleVec}(\mathbf{ctx}_{\mathsf{PKE.ek}}, \mathcal{R})$
8: $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$
9: $\mathbf{ek}_{\mathsf{PKE}} \leftarrow (\mathbf{seed}_{\mathsf{PKE.ek}}, \mathbf{s})$
10: **return** $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}})$

C.3 KEM

The formal procedures of HARE-KEM are specified in Algorithm 6, Algorithm 7, and Algorithm 8.

D Security Proof

D.1 Proof of IND-CPA Security

We prove Theorem 5 by a sequence of games. Let $b_1 = \mathbf{h}(1) \bmod 2$, $b_2 = w + b_1 \times w \bmod 2$ and $b_3 = w_r + b_1 \times w_r \bmod 2$. Let $w_0 = w_r$, $w_1 = w_e$, and $l = l_1 = n_1 \times n_2$.

Algorithm 4 PKE.Encrypt()

Input: Encryption key $\mathbf{ek}_{\mathsf{PKE}}$, message $\mathbf{m}$ and randomness θ
Output: Ciphertext $\mathbf{c}$

$\mathbf{seed}_{\mathsf{PKE.ek}} \leftarrow \mathbf{ek}_{\mathsf{PKE}}[0 : |\mathbf{seed}| - 1]$
$\mathbf{ctx}_{\mathsf{PKE.ek}} \leftarrow \mathsf{XOF.Init}(\mathbf{seed}_{\mathsf{PKE.ek}})$
$\mathbf{ctx}_{\mathsf{PKE.ek}}, \mathbf{h} \leftarrow \mathsf{SampleVec}(\mathbf{ctx}_{\mathsf{PKE.ek}}, \mathcal{R})$
$\mathbf{s} \leftarrow \mathbf{ek}_{\mathsf{PKE}}[|\mathbf{seed}| : |\mathbf{seed}| + |\mathbf{s}| - 1]$
$\mathbf{ctx}_{\theta} \leftarrow \mathsf{XOF.Init}(\theta)$
$\mathbf{ctx}_{\theta}, \mathbf{e} \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\theta}, \mathcal{R}_{w_e})$
$\mathbf{ctx}_{\theta}, \mathbf{r}_1 \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\theta}, \mathcal{R}_{w_r})$
$\mathbf{ctx}_{\theta}, \mathbf{r}_2 \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\theta}, \mathcal{R}_{w_r})$
$\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$
$\mathbf{v} \leftarrow \mathbf{m}\mathbf{G}_1 + \mathsf{Truncate}(\mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}, l_1)$
$\mathbf{v}_1 \leftarrow \mathbf{v}[0, l_2 - 1]$
$\mathbf{v}_2 \leftarrow \mathbf{v}[l_2, l_1 - 1]$
$\tilde{\mathbf{m}} \leftarrow \mathsf{Compress}_{\mathcal{C}_2}(\mathbf{v}_1)$
$\mathbf{c} \leftarrow (\mathbf{u}, \tilde{\mathbf{m}}, \mathbf{v}_2)$
return $\mathbf{c}$

Algorithm 5 PKE.Decrypt()

Input: Decryption key $\mathbf{dk}_{\mathsf{PKE}}$ and ciphertext $\mathbf{c}$
Output: Message $\mathbf{m}$

$\mathbf{ctx}_{\mathsf{PKE.dk}} \leftarrow \mathsf{XOF.Init}(\mathbf{dk}_{\mathsf{PKE}})$
$\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathbf{y} \leftarrow \mathsf{SampleFixedWeightVec}(\mathbf{ctx}_{\mathsf{PKE.dk}}, \mathcal{R}_w)$
$\mathbf{u} \leftarrow \mathbf{c}[0 : |\mathbf{u}| - 1]$
$\tilde{\mathbf{m}} \leftarrow \mathbf{c}[|\mathbf{u}| : |\mathbf{u}| + |\tilde{\mathbf{m}}| - 1]$
$\mathbf{v}_2 \leftarrow \mathbf{c}[|\mathbf{u}| + |\tilde{\mathbf{m}}| : |\mathbf{c}| - 1]$
$\tilde{\mathbf{v}} \leftarrow \mathsf{Decompress}_{\mathcal{C}_2}(\tilde{\mathbf{m}}) \parallel \mathbf{v}_2$
$\mathbf{m} \leftarrow \mathcal{C}_1.\mathsf{Decode}(\tilde{\mathbf{v}} - \mathsf{Truncate}(\mathbf{uy}, l_1))$
return $\mathbf{m}$

Algorithm 6 HARE-KEM.Keygen()

Input: None
Output: Encapsulation key $\mathbf{ek}_{\mathsf{KEM}}$ and decapsulation key $\mathbf{dk}_{\mathsf{KEM}}$

$\mathbf{seed}_{\mathsf{KEM}} \xleftarrow{\$} \mathcal{B}^{|\mathbf{seed}|}$
$\mathbf{ctx}_{\mathsf{KEM}} \leftarrow \mathsf{XOF.Init}(\mathbf{seed}_{\mathsf{KEM}})$
$(\mathbf{ctx}_{\mathsf{KEM}}, \mathbf{seed}_{\mathsf{PKE}}) \leftarrow \mathsf{XOF.GetBytes}(\mathbf{ctx}_{\mathsf{KEM}}, |\mathbf{seed}|)$
$(\mathbf{ctx}_{\mathsf{KEM}}, \sigma) \leftarrow \mathsf{XOF.GetBytes}(\mathbf{ctx}_{\mathsf{KEM}}, |\mathbf{K}|)$
$(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{HARE\text{-}PKE.Keygen}(\mathbf{seed}_{\mathsf{PKE}})$
$\mathbf{ek}_{\mathsf{KEM}} \leftarrow \mathbf{ek}_{\mathsf{PKE}}$
$\mathbf{dk}_{\mathsf{KEM}} \leftarrow (\mathbf{ek}_{\mathsf{KEM}}, \mathbf{dk}_{\mathsf{PKE}}, \sigma, \mathbf{seed}_{\mathsf{KEM}})$
return $(\mathbf{ek}_{\mathsf{KEM}}, \mathbf{dk}_{\mathsf{KEM}})$

Algorithm 7 HARE-KEM.Encaps($\mathbf{ek}_{\mathsf{KEM}}$)

Input: Encapsulation key $\mathbf{ek}_{\mathsf{KEM}}$
Output: Shared secret key $\mathbf{K}$ and ciphertext $\mathbf{c}_{\mathsf{KEM}}$

$\mathbf{m} \xleftarrow{\$} \mathcal{B}^{k/8}$
$\mathbf{salt} \xleftarrow{\$} \mathcal{B}^{|\mathbf{salt}|}$
$(\mathbf{K}, \theta) \leftarrow \mathsf{G}(\mathsf{H}(\mathbf{ek}_{\mathsf{KEM}}) \parallel \mathbf{m} \parallel \mathbf{salt})$
$\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{HARE\text{-}PKE.Encrypt}(\mathbf{ek}_{\mathsf{KEM}}, \mathbf{m}, \theta)$
$\mathbf{c}_{\mathsf{KEM}} \leftarrow (\mathbf{c}_{\mathsf{PKE}}, \mathbf{salt})$
return $(\mathbf{K}, \mathbf{c}_{\mathsf{KEM}})$

Algorithm 8 HARE-KEM.Decaps($\mathbf{dk}_{\mathsf{KEM}}, \mathbf{c}_{\mathsf{KEM}}$)

Input: Decapsulation key $\mathbf{dk}_{\mathsf{KEM}}$ and ciphertext $\mathbf{c}_{\mathsf{KEM}}$
Output: Shared secret key $\mathbf{K}'$

$\mathbf{ek}_{\mathsf{KEM}} \leftarrow \mathbf{dk}_{\mathsf{KEM}}[0 : |\mathbf{ek}_{\mathsf{KEM}}| - 1]$
$\mathbf{dk}_{\mathsf{PKE}} \leftarrow \mathbf{dk}_{\mathsf{KEM}}[|\mathbf{ek}_{\mathsf{KEM}}| : |\mathbf{ek}_{\mathsf{KEM}}| + |\mathbf{dk}_{\mathsf{PKE}}| - 1]$
$\sigma \leftarrow \mathbf{dk}_{\mathsf{KEM}}[|\mathbf{ek}_{\mathsf{KEM}}| + |\mathbf{dk}_{\mathsf{PKE}}| : |\mathbf{ek}_{\mathsf{KEM}}| + |\mathbf{dk}_{\mathsf{PKE}}| + |\sigma| - 1]$
$\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathbf{c}_{\mathsf{KEM}}[0 : |\mathbf{c}_{\mathsf{PKE}}| - 1]$
$\mathbf{salt} \leftarrow \mathbf{c}_{\mathsf{KEM}}[|\mathbf{c}_{\mathsf{PKE}}| : |\mathbf{c}_{\mathsf{PKE}}| + |\mathbf{salt}| - 1]$
$\mathbf{m}' \leftarrow \mathsf{HARE\text{-}PKE.Decrypt}(\mathbf{dk}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
$(\mathbf{K}', \theta') \leftarrow \mathsf{G}(\mathsf{H}(\mathbf{ek}_{\mathsf{KEM}}) \parallel \mathbf{m}' \parallel \mathbf{salt})$
$\mathbf{c}'_{\mathsf{PKE}} \leftarrow \mathsf{HARE\text{-}PKE.Encrypt}(\mathbf{ek}_{\mathsf{KEM}}, \mathbf{m}', \theta')$
$\mathbf{c}'_{\mathsf{KEM}} \leftarrow (\mathbf{c}'_{\mathsf{PKE}}, \mathbf{salt})$
$\bar{\mathbf{K}} \leftarrow \mathsf{J}(\mathsf{H}(\mathbf{ek}_{\mathsf{KEM}}) \parallel \sigma \parallel \mathbf{c}_{\mathsf{KEM}})$
if $\mathbf{m}' = \perp$ **or if** $\mathbf{c}'_{\mathsf{KEM}} \neq \mathbf{c}_{\mathsf{KEM}}$ **then**
    $\mathbf{K}' \leftarrow \bar{\mathbf{K}}$
end if
return $\mathbf{K}'$

Proof. We divide the proof into four stages.

(1) Remove the compression technique. Let $\mathsf{HARE\text{-}PKE}_0$ be our PKE scheme without the ciphertext compression, i.e., keeping $\mathbf{v}$ instead of $\tilde{\mathbf{m}}, \mathbf{v}_2$ as the ciphertext. Let Game_0 be the IND-CPA game for HARE-PKE and Game_1 be the IND-CPA game for $\mathsf{HARE\text{-}PKE}_0$. For any adversary $\mathcal{A}_1$ against HARE-PKE, we construct $\mathcal{A}_0$ to break the IND-CPA of $\mathsf{HARE\text{-}PKE}_0$. The procedures are as follows.

1. $\mathcal{A}_0$ receives the encryption key $\mathbf{ek}$ from $\mathsf{HARE\text{-}PKE}_0$ and sends it to $\mathcal{A}_1$.
2. $\mathcal{A}_0$ gets $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{1,\mathsf{CHOOSE}}(\mathbf{ek})$ and sends it to the challenger of $\mathsf{HARE\text{-}PKE}_0$.
3. The challenger randomly selects $b \xleftarrow{\$} \{0,1\}, \theta \xleftarrow{\$} \mathcal{B}^{|\theta|}$ and generates $\mathbf{c} \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Encrypt}(\mathbf{ek}, \mathbf{m}_b, \theta)$.
4. Let $(\mathbf{u}, \mathbf{v}) \leftarrow \mathbf{c}$, $\mathcal{A}_0$ calculates $\mathbf{v}_1 = \mathbf{v}[0, l_2 - 1]$, $\mathbf{v}_2 = \mathbf{v}[l_2, l_1 - 1]$, and $\tilde{\mathbf{m}} = \mathsf{Compress}_{\mathcal{C}_2}(\mathbf{v}_1)$. Then $\mathcal{A}_0$ sends the challenge ciphertext $(\mathbf{u}, \tilde{\mathbf{m}}, \mathbf{v}_2)$ to $\mathcal{A}_1$.
5. $\mathcal{A}_0$ outputs the guess bit b' received from $\mathcal{A}_1$.

The transformation from $(\mathbf{u}, \mathbf{v})$ to $(\mathbf{u}, \tilde{\mathbf{m}}, \mathbf{v}_2)$ is a publicly computable mapping requiring no secret information. Since $\mathcal{A}_0$ applies the exact compression defined in HARE-PKE, the challenge ciphertext provided to $\mathcal{A}_1$ is identically distributed to a real ciphertext in HARE-PKE. Thus, $\mathcal{A}_1$ has a perfectly identical view in the reduction as in the real IND-CPA game, which directly implies:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HARE\text{-}PKE}}(\mathcal{A}_1) = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HARE\text{-}PKE}_0}(\mathcal{A}_0). \tag{19}$$

(2) Replace the encryption key. We first define the game against $\mathsf{HARE\text{-}PKE}_0$. For the sake of simplicity, we use the vector $\mathbf{h}$ as part of the encryption key instead of the seed to generate it.

Game$_1(\lambda)$

1. $\mathbf{seed}_{\mathsf{PKE}} \xleftarrow{\$} \mathcal{B}^{|\mathbf{seed}_{\mathsf{PKE}}|}$
2. $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Keygen}(\mathrm{seed}_{\mathsf{PKE}})$
3. $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{0,\mathsf{CHOOSE}}(\mathbf{ek}_{\mathsf{PKE}})$
4. $b \xleftarrow{\$} \{0,1\}, \theta \xleftarrow{\$} \mathcal{B}^{|\theta|}$
5. $\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Encrypt}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{m}_b, \theta)$
6. $b' \leftarrow \mathcal{A}_{0,\mathsf{GUESS}}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
7. **return** $(b = b')$

Then we forget the decryption key $\mathbf{dk}_{\mathsf{PKE}}$, replace $\mathbf{s}$ of the encryption key with a random vector with parity b_2, and construct Game_2 with other steps unchanged.

$\mathbf{Game}_2(\lambda)$

1. $\mathbf{seed}_{\mathsf{PKE}} \xleftarrow{\$} \mathcal{B}^{|\mathbf{seed}_{\mathsf{PKE}}|}$
2. $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Keygen}(\mathrm{seed}_{\mathsf{PKE}})$
3. $\mathbf{s} \xleftarrow{\$} \mathbb{F}_{2,b_2}^n$
4. $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow ((\mathbf{seed}_{\mathsf{PKE.ek}}, \mathbf{s}), \mathbf{0})$
5. $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{0,\mathsf{CHOOSE}}(\mathbf{ek}_{\mathsf{PKE}})$
6. $b \xleftarrow{\$} \{0,1\}, \theta \xleftarrow{\$} \mathcal{B}^{|\theta|}$
7. $\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Encrypt}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{m}_b, \theta)$
8. $b' \leftarrow \mathcal{A}_{0,\mathsf{GUESS}}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
9. **return** $(b = b')$

Using $\mathcal{A}_0$, we construct a distinguisher $\mathcal{B}_1$ for 2-DQCSD-P as follows.

Adversary against 2-DQCSD-P($\mathbf{H}$,$\mathbf{s}$)

1. Compute $\mathbf{h}$ from $\mathbf{H} = (\mathbf{I}_n, \mathtt{rot}(\mathbf{h}))$
2. Set $\mathbf{ek}_{\mathsf{PKE}} \leftarrow (\mathbf{h}, \mathbf{s})$
3. Get $\mathbf{m}_0, \mathbf{m}_1 \leftarrow \mathcal{A}_{0,\mathsf{CHOOSE}}(\mathbf{ek}_{\mathsf{PKE}})$
4. Sample $b \xleftarrow{\$} \{0,1\}$ and $\theta \xleftarrow{\$} \mathcal{B}^{|\theta|}$
5. Compute $\mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Encrypt}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{m}_b, \theta)$
6. Get $b' \leftarrow \mathcal{A}_{0,\mathsf{GUESS}}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
7. **return** $(b = b')$

The only difference between Game_1 and Game_2 is the distribution of $(\mathbf{h}, \mathbf{s})$. Thus, the advantage ε of distinguishing Game_1 and Game_2 is precisely the advantage $\mathsf{Adv}_{\text{2-DQCSD-P}}$ of distinguishing the distribution of $(\mathbf{h}, \mathbf{s})$ from the random distribution.

(3) Replace the ciphertext. Then we construct Game_3 by replacing the ciphertext. Instead of sampling $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}$ with correct weight, we sample random vectors with expected parity.

$\mathbf{Game}_3(\lambda)$

1. $\mathbf{seed}_{\mathsf{PKE}} \xleftarrow{\$} \mathcal{B}^{|\mathbf{seed}_{\mathsf{PKE}}|}$
2. $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{HARE\text{-}PKE}_0.\mathsf{Keygen}(\mathrm{seed}_{\mathsf{PKE}})$
3. $\mathbf{s} \xleftarrow{\$} \mathbb{F}^n_{2,b_2}$
4. $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow ((\mathbf{seed}_{\mathsf{PKE.ek}}, \mathbf{s}), \mathbf{0})$
5. $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{0,\mathsf{CHOOSE}}(\mathbf{ek}_{\mathsf{PKE}})$
6. $b \xleftarrow{\$} \{0,1\}$
7. $\mathbf{e} \xleftarrow{\$} \mathbb{F}^n_2, (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathbb{F}^n_{2,w_r \bmod 2} \times \mathbb{F}^n_{2,w_r \bmod 2}$
8. $\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$
9. $\mathbf{v} \leftarrow \mathbf{m}_b\mathbf{G}_1 + \mathsf{Truncate}(\mathbf{s}\mathbf{r}_2 + \mathbf{e}, l_1)$
10. $\mathbf{c}_{\mathsf{PKE}} \leftarrow (\mathbf{u}, \mathbf{v})$
11. $b' \leftarrow \mathcal{A}_{0,\mathsf{GUESS}}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
12. **return** $(b = b')$

Using $\mathcal{A}_0$, we construct a distinguisher $\mathcal{B}_2$ for 3-DUQCSD-PT as follows.

Adversary against 3-DUQCSD-PT$(\mathbf{H}, (\mathbf{u}, \mathbf{v}))$

1. Compute $\mathbf{h}$ and $\mathbf{s}$ from $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathtt{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathtt{rot}(\mathbf{s}) \end{pmatrix}$
2. Compute $\mathbf{ek}_{\mathsf{PKE}} \leftarrow (\mathbf{h}, \mathbf{s})$
3. Get $(\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_{0,\mathsf{CHOOSE}}(\mathbf{ek}_{\mathsf{PKE}})$
4. Sample $b \xleftarrow{\$} \{0,1\}$
5. Compute $\mathbf{c}_{\mathsf{PKE}} \leftarrow (\mathbf{u}, \mathbf{m}_b\mathbf{G}_1 + \mathbf{v})$
6. Get $b' \leftarrow \mathcal{A}_{0,\mathsf{GUESS}}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}})$
7. **return** $(b = b')$

With

$$(\mathbf{u}, \mathbf{v}-\mathbf{m}_b\mathbf{G}_1)^\top = ((\mathbf{I}_n\ \mathbf{0}\ \mathtt{rot}(\mathbf{h}))\cdot(\mathbf{r}_1\ \mathbf{e}\ \mathbf{r}_2)^\top, \mathsf{Truncate}((\mathbf{0}\ \mathbf{I}_n\ \mathtt{rot}(\mathbf{s}))\cdot(\mathbf{r}_1\ \mathbf{e}\ \mathbf{r}_2)^\top, l)),$$

the only difference between Game_2 and Game_3 is whether the pair

$$\left(\begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathtt{rot}(\mathbf{h}) \\ \mathbf{0} & \mathbf{I}_n & \mathtt{rot}(\mathbf{s}) \end{pmatrix}, (\mathbf{u}, \mathbf{v} - \mathbf{m}_b\mathbf{G}_1)\right)$$

follows the 3-UQCSD-PT$(n, w_0, w_1, b_1, b_2, b_3, l)$ distribution or a uniform distribution with parity over $\mathbb{F}^{2n\times 3n}_{2,b_1,b_2} \times (\mathbb{F}^n_{2,b_3} \times \mathbb{F}^l_2)$. The structured and uniform inputs to $\mathcal{B}_2$ induce Game_2 and Game_3, respectively, for $\mathcal{A}_0$. Thus, the distinguishing advantage of $\mathcal{B}_2$ equals the absolute difference between the probabilities that these two games output 1.

(4) Conclusion. In Game_3, since $\mathbf{u}$ is computed independently of the challenge bit b and $\mathbf{v}$ is masked by a random vector $\mathbf{e}$, the advantage on Game_3 is 0.

With the previous proof chain, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HARE\text{-}PKE}}(\mathcal{A}_1) &= \mathsf{Adv}^{\mathsf{IND\text{-}CPA},\mathsf{Game}_1}_{\mathsf{HARE\text{-}PKE}_0}(\mathcal{A}_0) \\ &\leq \mathsf{Adv}_{\mathsf{2\text{-}DQCSD\text{-}P}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA},\mathsf{Game}_2}_{\mathsf{HARE\text{-}PKE}_0}(\mathcal{A}_0) \\ &\leq \mathsf{Adv}_{\mathsf{2\text{-}DQCSD\text{-}P}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{3\text{-}DUQCSD\text{-}PT}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA},\mathsf{Game}_3}_{\mathsf{HARE\text{-}PKE}_0}(\mathcal{A}_0) \\ &= \mathsf{Adv}_{\mathsf{2\text{-}DQCSD\text{-}P}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{3\text{-}DUQCSD\text{-}PT}}(\mathcal{B}_2).\end{aligned}$$

which completes the proof.

□

D.2 IND-CCA2 Security Bound

We derive the IND-CCA2 security bound of HARE-KEM based on the salted FO transformation [15,17,21]. We first recall the correctness definitions required by the transformation.

Definition 17 (δ-correct PKE [17]). *A* PKE *is δ-correct if:*

$$\mathbb{E}\left(\max_{\mathbf{m}\in\mathcal{M}} \Pr\left[\mathsf{PKE.Decrypt}(\mathbf{dk}_{\mathsf{PKE}}, \mathbf{c}_{\mathsf{PKE}}) \neq \mathbf{m} \mid \mathbf{c}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Encrypt}(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{m})\right]\right) \leq \delta. \tag{20}$$

where the expectation is taken over $(\mathbf{ek}_{\mathsf{PKE}}, \mathbf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.Keygen}()$.

Definition 18 (δ-correct KEM [17]). *A* KEM *is δ-correct if:*

$$\Pr\left[\mathsf{KEM.Decaps}(\mathbf{dk}_{\mathsf{KEM}}, \mathbf{c}_{\mathsf{KEM}}) \neq \mathbf{K} \;\middle|\; \begin{array}{l}(\mathbf{ek}_{\mathsf{KEM}}, \mathbf{dk}_{\mathsf{KEM}}) \leftarrow \mathsf{KEM.Keygen}(); \\ (\mathbf{K}, \mathbf{c}_{\mathsf{KEM}}) \leftarrow \mathsf{KEM.Encaps}(\mathbf{ek}_{\mathsf{KEM}})\end{array}\right] \leq \delta. \tag{21}$$

For HARE-PKE and HARE-KEM, the correctness parameter δ is bounded by the DFR analysis in Section 7. Then, by applying Theorem 13 from [17] to the salted FO transformation and substituting the IND-CPA bound from Theorem 5, we obtain the IND-CCA2 security bound stated in Theorem 6.