SLEEK: Sparse Quasi-Cyclic Codes with Closed-Form DFR Key Encapsulation

Alessandro Annechini, Alessandro Barenghi, Marco Gianvecchio and Gerardo Pelosi

Politecnico di Milano, Milan, Italy
Department of Electronics, Information and Bioengineering (DEIB)
{alessandro.annechini, marco.gianvecchio, alessandro.barenghi, gerardo.pelosi}@polimi.it

Abstract. Performance in terms of bandwidth and execution time, as well as decoding failures remain important challenges for code-based KEMs. This paper introduces SLEEK: *Sparse quasi-cycLic codEs with closEd-form DFR Key encapsulation*, a Key Encapsulation Mechanism (KEM) based on Quasi-Cyclic Moderate-Density Parity-Check (QC-MDPC) codes. The scheme employs a parallel bit-flipping decoder with three iterations and a closed-form estimate of the decoding-failure rate (DFR). The latter allows parameter selection with analytically bounded failure probability and supports the use of standard IND-CCA transformations.
We designed two variants of SLEEK. The first, SLEEK-LT, targets IND-CCA security for long-term keys, and supports QC-MDPC codes with different rates, allowing a tradeoff between public-key and ciphertext size. The second, SLEEK-E, targets IND-CPA security for ephemeral keys in settings requiring forward secrecy. While SLEEK-LT makes conservative choices and aims at security against active attackers, SLEEK-E relies on a novel QC key structure, for which we establish security equivalent to that of the original QC-MDPC construction under the same assumptions, while reducing key-generation cost. We explore a broad design space and provide parameter sets targeting 128-, 192-, and 256-bit security, together with an 80-bit configuration for constrained embedded platforms. We also optimize polynomial inversion and parallel bit-flipping decoding, the two dominant components of the implementation. We provide quantitative benchmarks of our optimized AVX2 implementation of SLEEK. For SLEEK-LT, ciphertexts are 4.9× smaller than those of the NIST standard choice, HQC, at the cost of a 47% increase in the runtime, and is 1.3× to 2.7× faster than BIKE. For SLEEK-E, execution time improves by 1.6× over HQC with a bandwidth usage reduced by 2.1×. Compared with Lean-BIKE, SLEEK-E is 5.3× faster and substantially reduces the impact of decoding failures on system reliability.

Keywords: Post-quantum cryptography · Code-based cryptography · QC-MDPC codes · Decoding failure rate (DFR) · Bit-flipping decoding

1 Introduction

The development of quantum computers and the improvements in solvers for discrete logarithms on elliptic curves [CXK+26] provide a significant push for a transition to post-quantum cryptography, which has been acknowledged by public institutions around the world [Uni22, Nat24b, Eur24, Eur25]. Among the available solutions, code-based cryptosystems [McE78, Nie86, BBC+18, BBC+20, ABC+20, ABB+23, GMA+25, AMAD+24] enjoy a high level of trust, stemming from the computational hardness of decoding random linear codes, which is well understood [BMvT78]. This maturity made code-based

Key Encapsulation Mechanisms (KEMs) interesting candidates throughout the NIST Post-Quantum Cryptography (PQC) Standardization Process [Nat25]. This has led NIST to select HQC, a code-based cryptosystem to differentiate the hard problems on which NIST standards rely on, as ML-KEM [Nat24a], the other selection, relies on hard problems from lattice theory. Code-based cryptosystems are built around two frameworks, the one by McEliece and Niederreiter [BMvT78], and the one by Alekhnovich [Ale03]. The latter does not require any properties from an efficiently decodable error correction code involved in the construction of the cipher, but yields larger ciphertexts and public keys than the former. The former, by contrast, requires error correction code which can be represented through a public parity check matrix which should be indistinguishable from a random one (in polynomial time). Only two families of error correction codes survived cryptanalytical scrutiny in this sense: binary Goppa codes and Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) codes. The former ones, proposed by McEliece in the original cryptosystem proposal, led to the Classic McEliece [ABC+20] specification. However, recent cryptanalytic progress has led to a sub-exponential time algorithm to retrieve the hidden Goppa code [Wei26], leading to controversies, and the retraction of the recommendation of the German BSI to employ the said cipher [BSI26]. The latter ones, QC-MDPC codes, present a significant resistance to distinguishing thanks to their complete lack of algebraic structure, bar for the quasi cyclicity feature, which was shown not to be reducing the decoding hardness per se [BMvT78, ABP26]. Furthermore, employing QC codes allows a significant reduction (MB to kB) of the public key size. The NIST contest has seen three candidates relying on quasi-cyclic and QC-MDPC codes surviving the first round of cryptanalytic scrutiny: HQC [GMA+25] which builds on Alekhnovich's framework and uses them to improve its efficiency, and was selected to become NIST FIPS 207 alongside the lattice-based ML-KEM [Nat24a]; BIKE [ABB+23] which was the runner-up against HQC up to the final selection, and LEDAcrypt-KEM [BBC+20]. The reason for selecting HQC over BIKE in the standardization process, despite its $\approx 3\times$ lower bandwidth requirement thanks to the smaller ciphertexts, was due to BIKE assurances on the probability of (security-threatening) decoding-failures being deemed insufficient [ACD+25], despite the recent work estimating failures on a proxy decoder [ALM+25]. Indeed, decoding failures in BIKE reveal (partial) information on the private key of the KEM, breaking the cipher in an active attacker scenario. To prevent such attacks (a notion captured by the IND-CCA security property) requires failures to happen with a probability lower than $2^{-\lambda}$,where λ is the security margin of the scheme [HHK17].

An alternative method to estimate reliably the decoding failure probability, commonly known as *Decoding Failure Rate* (DFR), was presented for two-iteration decoders in [ABP24], which, while practical, require larger parameters than the ones employed in BIKE. The recently proposed improvement in [ABPP26] extending the two-iteration DFR estimates to three iteration decoders, in combination with the result stating that picking any QC-MDPC code with a constant rate (in the code length) of the form $\frac{n_0-1}{n_0}$, $n_0 \in \mathbb{N}, n_0 > 1$, does not alter the hardness of the underlying problems [ABP26], open up a potential design space for efficient and compact KEMs.

In this work, we aim to introduce and describe techniques for the efficient realization of a compact KEM based on the Niederreiter framework, employing QC-MDPC codes, improving on the current state of the art represented by BIKE, and comparing favourably with the NIST-selected code based standard KEM HQC. Our work thus follows the same investigation direction which the proposal of BAT [FKPY22] took for lattice-based NTRU-like KEMs, while exploring the code-based design space.

Contributions. This work explores the extent to which QC-MDPC codes can support KEM designs that simultaneously offer competitive performance, reduced communication costs, and analytically bounded decoding-failure probabilities.

We present SLEEK, a QC-MDPC-based KEM, with two variants targeting IND-CCA

and IND-CPA security (SLEEK-LT and SLEEK-E), and perform a broad design-space exploration, deriving concrete parameter sets achieving 128-, 192-, and 256-bit security for long-term and ephemeral-key usage, together with an 80-bit configuration aimed at deployments on constrained platforms.

We optimize polynomial inversion in $\mathbb{Z}_2[x]/\langle x^p - 1\rangle$, constituting the most computationally intensive operation performed in the key generation phase, and we introduce an improved parallel bit-flipping decoder with respect to [ABPP26]. The new decoder reduces execution time while retaining the same closed-form decoding-failure-rate (DFR) analysis, enabling parameter selection with analytically bounded failure probability and supporting standard IND-CCA security transformations.

For ephemeral-key deployments, we propose a novel quasi-cyclic key structure and prove that its security is equivalent to that of the original QC-MDPC construction under the same assumptions. The new structure further reduces key-generation costs and forms the basis of an IND-CPA variant of the scheme.

Finally, we compare an optimized AVX2 implementation of our proposed scheme against HQC, BIKE, and Lean-BIKE. For SLEEK-LT, ciphertext sizes are up to 4.9× smaller than those of HQC, while KEM execution is between 1.3× and 2.7× faster than BIKE. For the IND-CPA variant, execution time improves by about 1.6× over HQC while reducing bandwidth usage by a factor of 2.1×. Compared with Lean-BIKE, the proposed construction achieves a 5.3× speedup while also improving reliability.

2 Background

Notation. We denote scalar values with lowercase Latin letters such as p, k, n, while vectors are denoted with lowercase boldface Latin letters, e.g., $\mathbf{w} \in \mathbb{F}_2^\ell$, $\ell > 1$. Italic uppercase Latin letters are reserved for matrices, e.g., $A \in \mathbb{F}_2^{\ell_1 \times \ell_2}$, $\ell_1, \ell_2 > 1$. A square circulant matrix $B \in \mathbb{F}_2^\ell$, $\ell > 1$, having each row equal to the vector obtained by applying a 1-position right-shift to the topmost one, is denoted as either $\mathbf{circ}(B[0,:])$ or $\mathbf{circ}(B[:,0])$, where $B[0,:]$ and $B[:,0]$ denote the first row and the first column, respectively, of the matrix. Given a row/column vector $\mathbf{w} = [v_0, v_1, v_2, \ldots, v_{\ell-1}] \in \mathbb{F}_2^\ell$, the corresponding square circulant matrix $\mathbf{circ}(\mathbf{w}) \in \mathbb{F}_2^{\ell \times \ell}$, $\ell > 1$, is also denoted as a polynomial $\mathsf{b}(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{\ell-1} x^{\ell-1} \in \mathbb{F}_2[x]/\langle x^\ell - 1\rangle$, with $b_i = v_i$, $0 \leq i < \ell$, using the isomorphism between the rings $(\mathbf{circ}(\mathbf{w}), +, \cdot)$, and $(\mathbb{F}_2[x]/\langle x^\ell - 1\rangle, +, \cdot)$, where, with a slight abuse of notation, the internal composition laws of the two algebraic structures, are denoted with the same symbols ($+$ and $\cdot$, respectively), even though they refer to either additions (multiplications, resp.) between circulant matrices over $\mathbb{F}_2$ or modular additions (multiplications, resp.) between polynomials. The number of non-zero entries for a binary vector $\mathbf{w} \in \mathbb{F}_2^\ell$, $\ell > 1$, is referred to as the (Hamming) weight of the vector and denoted as $\mathtt{wt}(\mathbf{w})$. Analogously, when considering square circulant matrices where each row/column includes a fixed number of non-zero entries their weight is denoted as $\mathtt{wt}(A)$. Uppercase bold Latin letters denote sets (e.g., $\mathbf{V}$). Sets including binary vectors having length $n > 1$ and weight t, $1 < t \leq n$ are denoted as $\mathbf{V}_t^n$. Sampling uniformly at random from a given domain or set is denoted as $\leftarrow$, while $:=$ denotes the definition of a symbol or the assignment (copy) of one variable value into another variable. We will denote as CSPRNG-$\mathbf{S}(\mathtt{seed})$ a Cryptographically Secure Pseudorandom Number Generator outputting uniform pseudorandom elements from the set $\mathbf{S}$ expanding a seed $\mathtt{seed}$. Finally, we will require a family of distinct cryptographic hashes, denoted as $\mathtt{Hash}_i$ identified by an integer suffix i. Such hashes will be obtained through domain separation from SHAKE [Nat15], truncating its output to the desired length.

2.1 Quasi-Cyclic MDPC Codes

A binary linear code is a subspace of $\mathbb{F}_2^n$ obtained as the image of a bijective linear map $\mathcal{C}[n,k] : \mathbb{F}_2^k \to \mathbb{F}_2^n$, $n, k \in \mathbb{N}^+$, with $k \leq n$, between any binary k-tuple, known as *information word*, and a binary n-tuple, known as *codeword*. The values n and k are known as the *length* and the *dimension* of the code, respectively, and their ratio $\frac{k}{n}$ defines the code *rate*. The set of all vectors in $\mathbb{F}_2^n$ that are orthogonal to the codewords in $\mathcal{C}[n,k]$ is known as *dual code*, $\mathcal{C}[n,k]^\perp$ and can be built via a generator matrix H composed by $n-k$ linearly independent vectors in $\mathbb{F}_2^n$, which is also known as a *parity-check matrix* of $\mathcal{C}[n,k]$. The quantity $r{=}n{-}k$ is known as *redundancy* of the code. For any $\mathbf{x} \in \mathbb{F}_2^n$, $\mathbf{s} = H\mathbf{x} \in \mathbb{F}_2^r$, is known as the *syndrome* of $\mathbf{x}$ through H. Being H a basis of $\mathcal{C}[n,k]^\perp$, every $\mathbf{c} \in \mathcal{C}[n,k]$ is s.t. $H\mathbf{c} = \mathbf{0}^{r\times 1}$, i.e., any codeword in $\mathcal{C}[n,k]$ has a null syndrome through H. A binary linear code $\mathcal{C}[n,k]$ is (v, w)-regular if it admits a parity-check matrix, where v and w are the Hamming weights of each column and each row, respectively and $\frac{n-k}{n} = \frac{v}{w}$.

A particular family of linear codes, employed in many post-quantum cryptosystems, is the one of Quasi-Cyclic (QC) codes.

Definition 1 (Quasi-cyclic Codes)**.** A linear code $\mathcal{C}[n,k]$, with $n = p\,n_0$, $k = p\,k_0$ (and redundancy $r = n - k = p\,r_0$, $r_0 = n_0 - k_0$) is said to be quasi-cyclic (QC) if its generator and parity-check matrices G and H are obtained tiling $p{\times}p$ circulant matrices on a $k_0{\times}n_0$ and a $r_0{\times}n_0$ grid, respectively.

Circulant matrices form a commutative ring isomorphic to $\mathcal{R} = \mathbb{F}_2[x]/\langle x^p - 1\rangle$, allowing for a compact representation and efficient arithmetic operations. A QC code with rate $\frac{n_0-1}{n_0}$ (i.e., with $r_0 = n_0 - k_0 = 1$) where the $p \times pn_0$ parity-check matrix has a fixed column weight v is (v,vn_0)-regular. QC Low Density Parity-Check (QC-LDPC) codes admit a parity-check matrix with column weight $v \in \mathcal{O}(\log(n))$ [AAC+25, Gal62], while QC Medium Density Parity-Check (QC-MDPC) codes admit parity-check matrices with column weight $v \in \mathcal{O}(\sqrt{n})$ [Til18]. A QC-MDPC matrix is thus described by n_0 circulant blocks (i.e., $H = [H_0\ H_1\ \ldots\ H_{n_0-1}]$) or, equivalently, by n_0 elements of $\mathcal{R}$, each with Hamming weight equal to v (i.e., $H = [\mathsf{h}_0(x)\ \mathsf{h}_1(x)\ \ldots\ \mathsf{h}_{n_0-1}(x)]$).

2.2 QC-MDPC Niederreiter PKE

The Public Key Encryption (PKE) scheme underlying SLEEK is equivalent to the one employed in QC-MDPC code-based KEMs such as BIKE [ABB+23] and LEDAcrypt [BBC+20], based on a template first proposed by Niederreiter [Nie86]. Figure 1 shows the triplet of procedures (PKE-KeyGen, Enc, Dec) describing such PKE.

The key generation algorithm (PKE-KeyGen) derives the private key, sk from a uniformly sampled λ-bit string $\mathtt{sdK}$ taken as input. The private key sk is a sparse QC-MDPC matrix $H := [H_0, \ldots, H_{n_0-1}] \in \mathbb{F}_2^{p\times n_0 p}$. The row-weight v of each circulant block H_i, with $0 \leq i < n_0$, is chosen as $v = \mathcal{O}(\sqrt{n})$ matching the definition of a moderate density parity check matrix. The public key, pk, is derived as a random-looking quasi-cyclic parity-check matrix $M := [M_0, \ldots, M_{n_0-2}, I] \in \mathbb{F}_2^{(n-k)\times n}$, denoted as the *systematic form* of H, where each block is obtained as $M_i := H_{n_0-1}^{-1} H_i$, $0 \leq i < n_0 - 1$.

The deterministic encryption algorithm (Enc) takes as input the public key, pk, and a plaintext $\mathbf{e} \in \mathbb{F}_2^{n_0 p}$ of fixed weight $\mathtt{wt}(\mathbf{e}) = t$, and computes the ciphertext as the syndrome $\mathbf{s} = M\mathbf{e}^\intercal$. The deterministic decryption (Dec) exploits the moderately sparse private key, $\mathsf{sk} := \{H\}$, to recover the error vector $\mathbf{e} \in \mathbb{F}_2^n$, $n = p\,n_0$, using an iterative bit-flipping decoding algorithm Decoder. In case of an unsuccessful decoding due to either structural properties of the code or a wrong input value as a ciphertext, i.e., a value that does not correspond to any possible plaintext (e.g., an error vector with weight $\neq t$), the decryption algorithm outputs the value $\perp$.

PKE-Keygen($\mathtt{sdK}$):

$H := \text{GENERATEH}(\mathtt{sdK})$
for $i := 0$ **to** $n_0 - 2$ **do**
  $M_i := H_{n_0-1}^{-1} H_i$
$\mathsf{pk} := \{[M_0 \mid \ldots \mid M_{n_0-2}]\}$
$\mathsf{sk} := \{H, H_{n_0-1}^{-1}\}$
return $(\mathsf{pk}, \mathsf{sk})$

Enc($\mathsf{pk}, \mathbf{e}$):

$\mathbf{s} := \mathbf{0} \in \mathbb{F}_2^p$
for $j := 0$ **to** $n_0 - 1$ **do**
  $\mathbf{s} := \mathbf{s} + M_j \, \mathbf{e}_j^{\mathsf{T}}$
$\mathbf{s} := \mathbf{s} + \mathbf{e}_{n_0-1}^{\mathsf{T}}$
return $\mathbf{s}$

Dec($\mathsf{sk}, \mathbf{s}$):

$\mathbf{s}' := H_{n_0-1} \, \mathbf{s}$
$\{\mathbf{e}, \mathtt{res}\} := \text{DECODER}(\mathsf{sk}, \mathbf{s}')$
if $\mathtt{res} = \mathtt{false}$ **then**
  $\mathbf{e} := \perp$
return $\mathbf{e}$

Figure 1: Niederreiter-style PKE instantiated with QC-MDPC codes.

The security of the PKE shown in Figure 1 relies on the computational hardness of the syndrome decoding problem and the codeword finding problem for random-looking quasi-cyclic (QC) code instances. In [ABP26], the authors provide evidence that, even when taking into account both the quasi-cyclic structure of the code (and consequentially the one of the associated parity-check matrix) and a code rate $k/n = (n_0 - 1)/n_0$, with $n_0 \geq 2$, the quasi-cyclic syndrome decoding problem (QC-SDP) and the codeword finding problem (QC-CFP) remain NP-hard (with the corresponding decision versions being NP-complete, and thus equivalent) as per random codes [BMvT78]. From an operational perspective, the SDP for binary linear codes requires finding a low-weight (row) vector $\mathbf{e} \in \mathbb{F}_2^n$ satisfying $A\mathbf{e}^{\mathsf{T}} = \mathbf{s}$, given a random-looking parity-check matrix $A \in \mathbb{F}_2^{(n-k)\times n}$ and a (column vector) syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$. The CFP can be stated similarly by taking $\mathbf{s} = \mathbf{0}$.

The most efficient approach to solve QC-SDP and QC-CFP instances is the one based on Information Set Decoding (ISD) techniques [Pra62, Ste88, Dum89, BJMM12, CHLT26]. Denoting as $\mathtt{ISD}(n, k, t)$ the concrete computational complexity of solving an SDP instance on a random linear code with length n, dimension k and error weight t, the quasi-cyclic structure of QC-MDPC codes introduces an additional speedup, due to the Decode One Out of Many (DOOM) [Sen11] strategy, which is equal to $\sqrt{p}$ for the QC-SDP and p for the QC-CFP. This implies that the complexity of a Message Recovery Attack (MRA) against the QC-MDPC Niederreiter PKE is $\mathtt{ISD}(n_0 p, (n_0 - 1)p, t)/\sqrt{p}$, and the complexity of a Key Recovery Attack (KRA) is $\mathtt{ISD}(n_0 p, p, n_0 v)/p$. The second result is obtained by interpreting the secret matrix H as a generator matrix of the dual code, having length $n_0 p$ and rate $\frac{1}{n_0}$, and solving the corresponding QC-CFP instance over such code.

2.3 Bit Flipping Decoding

The goal of the DECODER procedure is to retrieve the secret message, composed by an binary error vector $\mathbf{e}$ of length $p\,n_0$ and Hamming weight t, given its syndrome $\mathbf{s}$ through the private QC-MDPC matrix $H = [H_0, \ldots, H_{n_0-1}]$ (i.e., $\mathbf{s} = H\mathbf{e}^{\mathsf{T}}$). A simple yet effective decoder to solve such a task is the *parallel bit flipping decoder*, first proposed by Gallager [Gal62] and shown in Algorithm 4. The algorithm works by counting, for each column $H[:, j]$ $(0 \leq j \leq (p-1)\,n_0)$, the number of shared asserted bits between $H[:, j]$ and $\mathbf{s}$, with the result being an integer called $\mathtt{upc}$ (unsatisfied parity check counter) with values between 0 and v (i.e., the weight of $H[:, j]$). The rationale is that, if the matrix H is sparse

Algorithm 4: BF DECODER

Input: $\mathbf{s}$: $p\times 1$ syndrome,
  H:$p\times(p\,n_0)$ parity-check matrix.
Output: $\bar{\mathbf{e}}$: error vector estimate,
  $\mathtt{res}$: $\mathtt{true}$ (ok), $\mathtt{false}$ (fail).
Data: $\mathtt{iterMax}$: fixed no. of iterations.

$\bar{\mathbf{e}} := 0_{1\times(p\,n_0)}$
for $\mathtt{iter} := 0$ **to** $\mathtt{iterMax} - 1$ **do**
  $\mathtt{upc} := 0_{1\times p\,n_0}$ // vector of integers
  for j **from** 0 **to** $p\,n_0 - 1$ **do**
    $\mathtt{upc}[j] := \langle \mathbf{s}, H[:, j] \rangle$ // $\sum_i s_i \cdot h_{i,j}$
  $\mathtt{th} := \text{THRESHOLD}(\mathtt{iter})$
  for j **from** 0 **to** $p\,n_0 - 1$ **do**
    if $(\mathtt{upc}[j] \geq \mathtt{th})$ **then**
      $\bar{\mathbf{e}}_j := \bar{\mathbf{e}}_j \oplus 1$
      $\mathbf{s} := \mathbf{s} \oplus H[:, j]$
if $(\mathbf{s} = 0_{p\times 1} \wedge \mathtt{wt}(\bar{\mathbf{e}}) = t)$ **then**
  $\mathtt{res} := \mathtt{true}$
else $\mathtt{res} := \mathtt{false}$
return $\bar{\mathbf{e}}, \mathtt{res}$

enough, then the columns of H composing $\mathbf{s}$ (corresponding to the positions of asserted bits in $\mathbf{e}$) will have a high overlap with the set of asserted bits in $\mathbf{s}$, that is, a high $\texttt{upc}$. Therefore, the subsequent step assumes that the position of the asserted bits in $\mathbf{e}$ where the $\texttt{upc}$ exceeds a certain threshold $\texttt{th}$ are where the set bits in $\mathbf{e}$ are likely to be. The decoder thus flips the said positions in the error estimate vector $\bar{\mathbf{e}}$, update the syndrome variable $\mathbf{s}$ so that $H(\mathbf{e} \oplus \bar{\mathbf{e}}) = \mathbf{s}$ holds. If the syndrome becomes filled with zeros, then $H(\mathbf{e} \oplus \bar{\mathbf{e}})^\intercal = 0$, meaning that $\mathbf{e} = \bar{\mathbf{e}}$.

An important issue of iterative decoders, such as the one shown in Algorithm 4, is that they exhibit a non-zero Decoding Failure Rate (DFR). While this constitutes a simple reliability issue in the case of ephemeral keys and IND-CPA security, decoding failures pose a security threat for long term keys, as active attackers are able to gather information about the private key with even a single decoding failure [GJS16, WWW23]. For this reason, a requirement for IND-CCA secure cryptosystem is that the decryption failure rate (which in this case coincides with the DFR of the underlying decoder) must be lower than $2^{-\lambda}$, where λ is the bit security level [HHK17]. Therefore, a factor to take into account when choosing the decoder to employ as the Decoder procedure is the feasibility of a closed-form DFR analysis. Both cryptosystems BIKE [ABB+23] and LEDAcrypt [BBC+20] employ a parallel bit flipping decoder during decryption: the former uses $\texttt{iterMax} = 7$ and a Threshold function depending also on the Hamming weight of $\mathbf{s}$, while the latter uses $\texttt{iterMax} = 2$. The choice of $\texttt{iterMax}$ yields a tradeoff between public key and ciphertext size, decoder runtime, and feasibility of a decoding failure rate analysis. Performing a large number of iterations, as in BIKE, allows for smaller block sizes p, and thus more compact public keys and ciphertexts, at the cost of a more complex decoding procedure and the absence of a reliable DFR estimate [ACD+25]. Setting $\texttt{iterMax} = 2$, as in LEDAcrypt, leads to a simpler decoding failure analysis, but forces the size of public keys and ciphertext to be around 2× larger than the ones of BIKE. In our proposed KEM SLEEK, we exploit the DFR analysis in [ABPP26] by setting $\texttt{iterMax} = 3$, and choosing code parameters that guarantee a sufficiently low DFR. With this choice, we simultaneously obtain an efficient decryption procedure and competitive public key/ciphertext sizes.

3 SLEEK-LT Cryptosystem

This section presents SLEEK-LT, an IND-CCA KEM built from the QC-MDPC Niederreiter PKE introduced in the previous section. Section 3.1 describes the scheme, while Section 3.2 covers its implementation, including polynomial multiplication as well as improved polynomial inversion and bit-flipping decoding algorithms.

3.1 Scheme Design and Security

SLEEK-LT is defined by the triple of primitives (KeyGen, Encaps, Decaps) shown in Figure 2. The construction we employ to turn the PKE shown in Figure 1 is a variation of the Fujisaki–Okamoto (FO) transformation [FO99, FO13] similar to the one employed in BIKE and LEDAcrypt-KEM, save for three design choices aimed at improving either the cryptosystem security or performance (highlighted in blue).

An OW-CPA secure PKE scheme is such that an adversary, given the encryption of a randomly chosen plaintext and the public key by which it was encrypted, cannot recover the plaintext in polynomial time, while a correct PKE scheme is such that, for any plaintext $\mathbf{m}$, $\text{Dec}(\mathsf{sk}, \text{Enc}(\mathsf{pk}, \mathbf{m})) = \mathbf{m}$. The FO is a widely popular approach to convert a passively secure, correct (OW-CPA) PKE scheme into a KEM providing indistinguishability against a chosen-ciphertext attack (IND-CCA), under the technical assumption that it is efficient to sample from a uniform distribution over the PKE messages. Given that a large amount of post-quantum PKEs are not correct, Hofheinz, Hövelmanns, and Kiltz

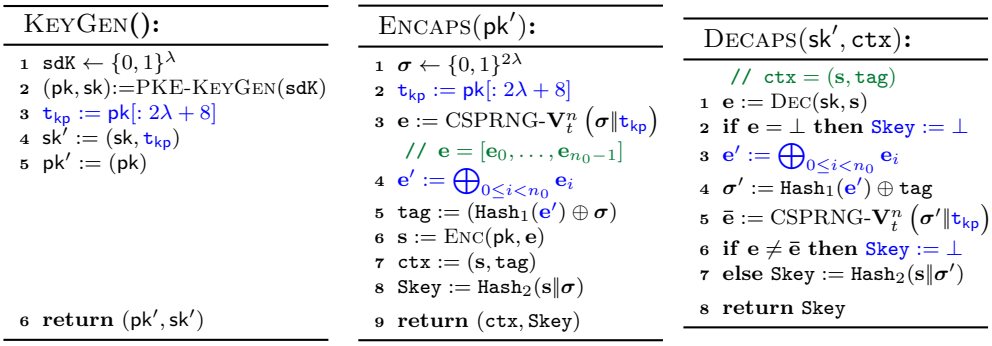


Figure 2: SLEEK-LT(KeyGen, Encaps, Decaps).

(HHK) [HHK17, HHK21] analyzed the FO transform and proved that it still yields an IND-CCA KEM even if starting from a OW-CPA secure PKE with a bounded error probability. Indeed, if the decryption failure rate is lower than $2^{-\lambda}$ (a condition denoted as δ-*correctness*), and the PKE satisfies the OW-CPA and input samplability conditions, the KEM resulting from the FO transform is IND-CCA. In our case, the OW-CPA security of SLEEK-PKE is ensured by a reduction of the QC-SDP and the QC-CFP to the message and key recovery attacks [ABP26], under the assumption (which underlies BIKE too) that recovering QC-MDPC sparse parity check matrix is computationally hard. To fulfill the δ-correctness condition for the SLEEK-PKE we combine two elements: the DFR analysis for the decoder we employ made in [ABPP26], which quantifies the average probability of a decoding failure (thus, a correctness failure) where the average is taken over the messages and keypairs, and a strategy to force the pseudorandom generation of the PKE message.

Operatively, the FO transform acts as the composition of two transforms [HHK17]: the T-transform, and one out the four possible choices for the U-transform. The T-transform is employed to turn a randomized OW-CPA PKE into a deterministic PKE, and to enforce the *rigidity* property [BP18], i.e., $\text{Enc}(\mathsf{pk}, \text{Dec}(\mathsf{sk}, \mathbf{c})) = \mathbf{c}$ for all valid ciphertexts. As a consequence of the PKE employed in SLEEK being rigid and deterministic, we can omit the application of the T-transform [HHMS25]. The U-transform starts from a rigid, deterministic OW-CPA PKE and turns it into an IND-CCA KEM in two steps. The first one turns the PKE into a KEM by uniformly randomly sampling its input message $\mathbf{m}$. The second step in the U-transform determines how to derive the session key `Skey`. Depending on the U-transform variant, this is done either hashing the PKE input $\mathbf{m}$ alone (denoted with the m subscript to U), or concatenated with the PKE ciphertext (no subscripts to U). The second step also needs to take care of the fact that the PKE decryption may fail: the U transform either reports a failure outputting a dedicated failure symbol $\perp$ (*explicit rejection*, denoted by a $\perp$ superscript to U), or it generates deterministically from the ciphertext and a dedicated λ bits secret string a pseudorandom `Skey` (*explicit rejection*, denoted by a $\not\perp$ superscript). We choose to apply $\mathsf{U}^{\not\perp}$ (explicit rejection and `Skey` derivation from both the message and the ciphertext). The rationale for our choice is twofold: recent works analyzing the so-called *binding properties* [JZM19] (i.e., resistance against attackers beyond the IND-CCA model) show that a large number of these [KMSW26] is achieved by explicitly-rejecting FO built KEMs. Furthermore, explicit rejection allows to avoid the storage of the additional secret string employed to generate the Decaps output in case of a failure, and the hash computation (which should always be performed to avoid timing leakage). This in turn reduces the attack surface against side-channel attacks [XIU+21].

SLEEK-LT is thus obtained applying the $\mathsf{U}^{\not\perp}$ to SLEEK-PKE, and applying two further tweaks, one to allow the verification of the sampling of the PKE message ((lines 1–3 in Encaps, lines 3–5 in Decaps), and the other to provide a simple ward against multikey attacks. The rationale of the first tweak, which first appeared in [BBC+20], and

was subsequently adopted by BIKE [ABB+23] is the following: the PKE input, i.e., the error vector $\mathbf{e}$ is sampled employing pseudorandomness coming from the expansion in a CSPRNG of a seed $\boldsymbol{\sigma}$. To allow the verification of such a process taking place, in the ENCAPS algorithm, a 2λ-bit value, $\mathtt{tag}$ is generated by masking $\boldsymbol{\sigma}$ with the digest of $\mathbf{e}$ ($\mathbf{e}'$ in our variant) via $\mathtt{xor}$. The DECAPS procedure, after retrieving the alleged value of $\mathbf{e}$ (and thus $\mathbf{e}'$ in our variant) via PKE DEC, unmasks $\mathtt{tag}$ obtaining the alleged value of $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$, and attempts to re-generate the same $\mathbf{e}$ expanding $\boldsymbol{\sigma}'$. If it succeeds, the DECAPS procedure can be confident that $\mathbf{e}$ was indeed pseudorandomly sampled, or an attacker managed to find a preimage for the CSPRNG [BBC+20], thus ensuring that the best probability of an attacker trying to make the DEC primitive fail is the one of trying to do so by uniformly picking errors to be decoded, matching the notion of DFR from [ABPP26]. In SLEEK-LT, we substitute $\mathbf{e}$ with $\mathbf{e}' = \bigoplus_{0 \le i < n_0} \mathbf{e}_i$ (where $\mathbf{e}_i$ is the i-th p-sized block of $\mathbf{e}$) as the input of the hash function, with the goal of reducing the size of the object to be hashed and thus improving performance. We now show that the security of our construction is not altered by this substitution, by proving that for an attacker finding $\mathbf{e}'$ is no easier than finding $\mathbf{e}$.

Lemma 1. *Finding $\mathbf{e}'$ has the same computational hardness of finding $\mathbf{e}$.*

Proof. In the parameter regime where the QC-MDPC based Niederreiter PKE is employed (i.e., $t \ll n_0 p$), we have that the probability that $\mathtt{wt}(\mathbf{e}') = t$ is $\Omega(1)$. This implies that, with high probability, $\mathrm{Supp}(\mathbf{e}_i) \subset \mathrm{Supp}(\mathbf{e}')$ for every $0 \le i < n_0$. To recover $\mathbf{e}$, it is thus sufficient to perform a single iteration of Information Set Decoding (ISD), selecting as a valid information set a total of k coordinates lying outside $\mathrm{Supp}(\mathbf{e}')$ in each block. Even if $\mathtt{wt}(\mathbf{e}') < t$, the probability of selecting a valid information set is still $1 - o(1)$. □

Finally, we employ an efficient and effective countermeasure against multikey attacks [WWW23]. This is done in BIKE by hashing the entire public key $\mathtt{pk}$, and concatenating its 2λ-bit digest to the input of the CSPRNG. Since the OW-CPA security of SLEEK PKE already relies on the indistinguishability of QC-MDPC from random QC codes, save for the parity of the Hamming weight of its circulant blocks, we have that the first b bits of the public key have at least $b - 1$ bits of entropy: we can therefore safely use the first $2\lambda + 8$ bits of the public key (instead of its hash) to avoid collisions in $\mathtt{t}_{\mathtt{kp}}$ between different public keys, and thus to thwart multikey attacks.

3.2 Efficient Implementation

In this section we discuss the efficient implementation of multiplications and multiplicative inverses in $\mathcal{R} = \mathbb{F}_2[x]/\langle x^p - 1\rangle$, along with a novel, optimized parallel bit flipping decoder, on $\mathtt{x86_64}$ platforms making use of the Intel AVX2 instruction set extension.

3.2.1 Polynomial Modular Multiplications over $\mathcal{R}$

Fast binary polynomial multiplications are a crucial component for efficiently computing the SLEEK-PKE algorithms shown in Fig. 1, thanks to the isomorphism between the ring of circulant matrices in $\mathbb{F}_2^{p \times p}$ and the ring of polynomials in $\mathcal{R} = \mathbb{F}_2[x]/\langle x^p - 1\rangle$. Given the length of the operands at hand, we employ the recursive Karatsuba [Kar63] algorithm. In the recursion steps where the size of the operands exceeds 512 bits, we employ the method proposed in [DG19] where the authors show that padding with zeroes the operands of the multiplication up to a length which is a power of two allows for the application of the Karatsuba technique through a remarkably regular code pattern which yields significant performance benefits. When the operand size reaches 512 bits, we switch to a different strategy. The AVX2 ISA extensions include the so-called *carry-less multiplication* instruction, introduced as the PCLMULQDQ in 2010 in Intel's Westmere

Algorithm 8: POLYNOMIAL INVERSE WITH FERMAT'S LITTLE THEOREM

Input: $\mathsf{h}(x)$: element of $\mathbb{F}_2[x]/\langle x^p - 1\rangle$ to be inverted.
Output: $\mathsf{res}(x)$: multiplicative inverse of $\mathsf{h}(x)$.

1 $\mathtt{exp} \leftarrow$ BINENCODING$(p-2)$ // $[\mathtt{exp}_0, \ldots, \mathtt{exp}_i, \ldots], \mathtt{exp}_i \in \{0,1\}, 0 \leq i < \lceil \log_2(p) \rceil$
2 $\mathsf{c}(x) \leftarrow \mathsf{h}(x)$
3 $\mathsf{res}(x) \leftarrow \mathsf{h}(x)$ // since $p-2$ is odd, we have $\mathtt{exp}_0 = 1$
4 **for** $i \leftarrow 1$ **to** $\lceil \log_2(p-2) \rceil - 1$ **do**
5 $\quad \mathsf{c}(x) \leftarrow \mathsf{c}(x) \cdot \mathsf{c}(x)^{2^{2^{i-1}}}$
6 $\quad$ **if** $\mathtt{exp}_i = 1$ **then** $\mathsf{res}(x) \leftarrow \mathsf{res}(x)^{2^{2^i}} \cdot \mathsf{c}(x)$
7 $\mathsf{res}(x) \leftarrow \mathsf{res}(x)^2$ // for the least significant zero-bit in the binary representation of $2^{p-1}-2$
8 **return** $\mathsf{res}(x)$

architecture, and later also adopted by ARM [GK10b, GK10a, TFA+11b, TFA+11a, SF12, Cho16, BBC+20, GdFAB19, DG19, DGK20b, DGK20a, DGK21, CCK21]. The PCLMULQDQ instruction performs a $64 \times 64 \rightarrow 128$-bit binary polynomial multiplication in constant time: two degree 63 operands are multiplied, and the degree-126 product is stored in an `xmm` register (128-bit wide). In [Mon05, FH07, CÖ09], the authors proposed tailored improvements to Karatsuba's method for small operands, up to nine 64b words, and [SF12] realized them using PCLMULQDQ as the base case. We thus switch to this multiplication technique once the size of the operands is reduced to less than nine 64b words. Since the operand size is fixed, all the macro-levels of recursion are implemented in constant time, without conditional branches depending on the value of the operands. Finally, after a multiplication of two p-bit polynomials is computed, the $2p$-bit result needs to be reduced modulo $(x^p - 1)$. Given the peculiar shape of the modulus $(x^p - 1)$, the modular reduction amounts to (`xor`-)adding the coefficients of the monomials between x^{2p-1} and x^p to the ones of the monomials between x^{p-1} and x^0 component-wise.

3.2.2 Polynomial Inverse Computation over $\mathcal{R}$

The KEYGEN procedure derives the public key $M = [\mathsf{m}_0(x), \ldots, \mathsf{m}_{n_0-2}(x), 1]$ from the private, sparse parity check matrix $H = [\mathsf{h}_0(x), \ldots, \mathsf{h}_{n_0-1}(x)]$ as $\mathsf{m}_i(x) = \mathsf{h}_{n_0-1}(x)^{-1}\,\mathsf{h}_i(x)$, $0 \leq i < n_0$. An efficient approach to compute $\mathsf{h}_{n_0-1}(x)^{-1}$ is based on Fermat's little theorem, stating that $\mathsf{h}(x)^{-1} = \mathsf{h}(x)^{2^{p-1}-2}$ [BP20]. The state-of-the-art procedure for computing $\mathsf{h}(x)^{2^{p-1}-2}$, first described in [DGK21], is reported in Alg. 8. Since the binary representation of the exponent to which $\mathsf{h}(x)$ should be raised, $2^{p-1} - 2$ is composed by $p-2$ ones followed by a zero, i.e., $2^{p-1} - 2 = \mathtt{11\ldots10}_{\mathsf{bin}}$, the working principle of Alg. 8 is to perform the analogous of a right-to-left double-and-add procedure applied to *the length of the sequence of* 1*s* in the binary representation of $2^{p-1} - 2$, to the end of getting a sequence which is $p-2$ ones long.

Alg. 8 involves just two modular operations on polynomials: iterated squaring and multiplication. Since iterated squaring can be efficiently performed by means of zero-interleaving or bitwise permutations [BP20, DGK21], modular multiplications are the only arithmetic operation whose complexity is not linear in p, therefore constituting the bottleneck of the execution. Alg. 8 requires $(\lceil \log_2(p-2) \rceil - 1) + (\mathtt{wt}(p-2) - 1)$ multiplications, with the second addend being due to the conditional multiplication performed in Line 8. This number corresponds to the amount of additions employed to compute $p-2$ with a double and add strategy starting from 1. While efficient, the performance of this approach heavily depends on the specific value of p. The BIKE design team [ABB+23] tackled this issue by selecting values of p such that $p-2$ has a low hamming weight (e.g., $p = 12323$, with $\mathtt{wt}(p-2) = 4$). Likewise, LEDAcrypt-KEM specification [BBC+20] shows that the search for a suitable value for p may require a $5 - 10\%$ increase in the block size. In both cases, the relation between the execution time of Alg. 8 and the Hamming weight of p

forces a compromise between a slower key generation procedure and a larger public key and ciphertext. We propose a technique based on *optimal addition chains* that removes the need for such a trade-off, allowing for the selection of the smallest possible p during parameter tuning regardless of its Hamming weight. An addition chain for a integer k is a strictly increasing sequence of l integers $q_0, q_1, \ldots, q_{l-1}$ such that: *1)* $q_0 = 1$, *2)* $q_{l-1} = k$, and *3)* for every element q_m, with $1 \leq m \leq l-1$, there are two elements q_i, q_j, with $i, j < m$, such that $q_m = q_i + q_j$. Observe that Alg. 8 computes an addition chain on the length of the runs of ones, which corresponds to a right-to-left double-and-add algorithm computing $p-2$. For many values of k, the addition chain with the shortest length (which may not be unique) is **not** the one defined by the double-to-add algorithm (either left-to-right or right-to-left). Since p is a fixed public parameter, we can improve on Alg. 8 by selecting a (fixed) sequence of multiplications according to the shortest addition chain composing $p-2$. Our implementation of such strategy works as follows: given the shortest addition chain $q_0, q_1, \ldots, q_{l-1}$ for $p-2$, the algorithm computes a sequence of polynomials $\mathsf{c}_m(x)$ $(0 \leq m \leq l-1)$ where $\mathsf{c}_0(x) = \mathsf{h}(x)$ and $\mathsf{c}_m(x) = \mathsf{c}_\mathsf{i}(x)^{2^{q_j}} \cdot \mathsf{c}_\mathsf{j}(x)$, where $q_i + q_j = q_m$. In this way, we can compute the polynomial $\mathsf{c}_{l-1}(x) = \mathsf{h}(x)^{2^{p-2}-1}$ (and thus $\mathsf{c}_{l-1}(x)^2 = \mathsf{h}(x)^{2^{p-1}-2} = \mathsf{h}(x)^{-1}$) using the smallest possible amount of multiplications.

3.2.3 A Novel QC-MDPC Parallel Bit-flipping Decoder

In SLEEK, we employ a variation of the parallel bit flipping decoder with $\texttt{iterMax} = 3$, balancing correction efficiency and performance while being able to reliably estimate its decoding failure rate [ABP24, ABPP26]. Such decoder, that we denote as SLEEKDEC, is shown in Algorithm 9. We first discuss how to efficiently implement the bit flipping decoder operations, and then we introduce our design optimization (highlighted in blue in Algorithm 9).

Algorithm 9: SLEEKDEC

Input: $\mathbf{s}$: $p\times 1$ syndrome,
H:$p\times(p\,n_0)$ parity-check matrix,
$H^{-1}_{n_0-1}$: inverse of the last block.
Output: $\bar{\mathbf{e}}$: error vector estimate,
res: `true` (ok), `false` (fail).
Data: `iterMax`: fixed no. of iterations.

1 $\bar{\mathbf{e}} := 0_{1\times(p\,n_0)}$
2 **for** iter $:= 0$ **to** iterMax $- 1$ **do**
3 **if** *(*iter$<$iterMax-1*)* **then** n $:= p\,n_0$
4 **else** n $:= p\,(n_0 - 1)$
5 upc $:= 0_{1\times \mathtt{n}}$ // vector of integers
6 **for** j **from** 0 **to** n $- 1$ **do**
7 upc$[j] := \langle \mathbf{s}, H[:, j]\rangle$ // $\sum_i s_i \cdot h_{i,j}$
8 th $:=$ THRESHOLD(iter)
9 **for** j **from** 0 **to** n $- 1$ **do**
10 **if** *(*upc$[j] \geq$ th*)* **then**
11 $\bar{\mathbf{e}}_j := \bar{\mathbf{e}}_j \oplus 1$
12 $\mathbf{s} := \mathbf{s} \oplus H[:, j]$
13 $\bar{\mathbf{e}} := \bar{\mathbf{e}} \oplus [0_{1\times p\,(n_0-1)} \mid (H^{-1}_{n_0-1}\mathbf{s})^{\mathrm{T}}]$
14 **if** *(*wt$(\bar{\mathbf{e}}) = t$*)* **then** res $:=$ `true`
15 **else** res $:=$ `false`
16 **return** $\bar{\mathbf{e}}$, res

Decoder Implementation. The most computationally demanding operation is computing the vector of upc values (lines $6-7$). The optimal approach for this task is to exploit the circulant structure of the blocks composing H. Such structure allows the upc vector in each block $0 \leq i \leq n_0-1$ to be computed as $\mathsf{upc}_i = \sum_{z\in \mathrm{Supp}(\mathsf{h}_i(x))} \varphi\left(\mathsf{s}(x)\cdot x^{-z}\right)$, where $\mathrm{Supp}(\mathsf{h}_i(x))$ denotes the set of asserted terms in the polynomial associated with the circulant block H_i, $\mathsf{s}(x)$ is the polynomial obtained from the binary coefficients of $\mathbf{s}$, and φ maps bijectively any polynomial in $\mathbb{F}_2[x]/\langle x^p+1\rangle$ into $\mathbb{Z}^p$ by associating the i-th coefficient of a polynomial with the i-th entry of the corresponding vector and storing in it the integer value 1 if the coefficient is non-zero, the integer value 0 otherwise. In other words, the vector upc_i corresponds to the sum over the integers of v circulant shifts of the syndrome vector. We therefore analyze the most efficient strategies for: *1)* computing syndrome-rotations in a constant-time manner, and *2)* computing the sum of v binary vectors over $\mathbb{Z}^p$.

Constant-time Vector Rotations. To perform this operation in a constant-time manner, we employ a Barrel shifter [Cho16], rotating the syndrome vector by 2^i positions for each i

between $\lfloor \log_2(p) \rfloor$ and 0, and committing the rotated vector only if the i-th least significant bit of the binary representation of `shiftAmt` is 1. In the first iterations, where the shift amount 2^i is greater than or equal to the size of an AVX2 register (256 bits), we implement the rotations by loading 256-bit chunks in an AVX2 register, and conditionally committing them in the correct position (we refer to this part of the procedure as `rotateBig`). In the last iterations, we perform shifts smaller than 256 bits by loading consecutive chunks of 256 bits and computing the rotation within the AVX2 registers (we refer to this part of the procedure as `rotateSmall`). One improvement, proposed in [CCK21], replaces the `rotateBig` part of the Barrel shifter by interpreting the binary vector as a $b \times b$ binary matrix (where $b^2 \approx p$), transposing it, and performing circulant rotations on the rows of this matrix by $\frac{2^i}{b}$ bits before inverting the transposition. In [CCK21], this rotation strategy is tested only for $p = 12323$ and $p = 24659$: in our parameter exploration, we found this approach to be beneficial for $p \geq 10000$, while for smaller block sizes the standard `rotateBig` procedure yields better performance figures.

Efficient Summation over $\mathbb{Z}^p$. The sum of v binary vectors can be efficiently computed in a bitsliced fashion, by implementing a chain of $\lceil \log_2(v) \rceil$ half-adders and computing v bitwise additions [Cho16]. One improvement on this approach, described in [CCK21], relies on a technique for computing the vector `upc` based on the Boyar-Peralta [BP05] design of the Hamming weight circuit. While this design yields the provably optimal circuit in terms of `and` gate count, it also significantly reduces the total number of required Boolean gates compared to previous approaches, at the cost of a small increase in memory footprint. We also tested the Hamming weight circuit design presented in [ABE+25], which provides approximately a further 10% reduction in the gate count with respect to [BP05]. This approach, however, requires a large number of intermediate results, and exhibits worse performance than the Boyar-Peralta design on our platform.

Novel Bit Flipping Decoder. To reduce the amount of expensive `upc` computations, we propose a modification to the parallel bit flipping decoder for SLEEK, highlighted in blue in Alg. 9. At any point during the decoding process, if the discrepancies remaining between $\mathbf{e}$ and $\bar{\mathbf{e}}$ all lie within the *last* p positions, we have that $\mathbf{s} = H(\mathbf{e} \oplus \bar{\mathbf{e}})^\intercal = H_{n_0-1}(\mathbf{e}_{n_0-1} \oplus \bar{\mathbf{e}}_{n_0-1})^\intercal$ where $\mathbf{e}_{n_0-1}$ and $\bar{\mathbf{e}}_{n_0-1}$ denote the last p positions of $\mathbf{e}$ and $\bar{\mathbf{e}}$, respectively. This implies that the correct value of $\mathbf{e}_{n_0-1}$ can be found as $\mathbf{e}_{n_0-1} = \bar{\mathbf{e}}_{n_0-1} \oplus (H^{-1}_{n_0-1}\mathbf{s})^\intercal$. Since this operation is faster than computing one block of the `upc` vector, and since $H^{-1}_{n_0-1}$ is a value pre-computed during the key generation procedure, we can skip the calculation of $\texttt{upc}_{n_0-1}$ entirely during the last iteration. Indeed, during the last iteration, the bit flipping decoder is assuming that all performed flipping decisions are correct, meaning that after $p\,(n_0 - 1)$ flipping decisions the remaining discrepancies are all in the last block. In addition to the performance improvement lead by this update, we point out that also the DFR benefits from this approach, as we are removing the possibility of failures caused by incorrect flips in the last block. For this reason, such modification to the decoder behaviour does not alter the validity of the DFR analysis from [ABPP26, ABP24], which can still be used as a conservative estimation for the SLEEK decoder.

4 SLEEK-E Cryptosystem

In this section, we describe SLEEK-E. Given the expected ephemeral key use of SLEEK-E, willing to reduce the keypair generation time, we introduce a class of QC codes, that we denote as *Bivariate QC-MDPC* (BQC-MDPC) codes, and listing the advantages led by their definition. Then, we describe how to efficiently integrate BQC-MDPC codes in the Niederreiter framework, leading to the SLEEK-E design, and we analyse the complexity of message and key recovery attacks specialized for this class of codes.

4.1 Bivariate QC-MDPC Codes

In the key generation procedure of the QC-MDPC based Niederreiter PKE, the most computationally heavy operation is deriving the multiplicative inverse of an element of the ring $\mathcal{R} = \mathbb{F}_2[x]/\langle x^p - 1\rangle$. The complexity of the inversion algorithm is superlinear in p, as the procedure requires $\mathcal{O}(\log_2(p))$ polynomial multiplications between elements in $\mathcal{R}$ (the details of the procedure are in Section 3.2.2). To reduce p without altering the length and dimension of the employed codes, we introduce the following class of QC codes.

Definition 2 (Bivariate Quasi-cyclic Codes)**.** A quasi-cyclic code is said to be *bivariate* quasi-cyclic (BQC) if its parity-check matrix H is composed by a pattern of $r_0 \times n_0$ blocks, where each block is obtained by tiling $\widetilde{r_0}$ circulant matrices (equiv. polynomials in $\mathcal{R}$) in a $\widetilde{r_0} \times \widetilde{r_0}$ circulant pattern.

From an arithmetic standpoint, the blocks of H form a commutative ring isomorphic to the polynomial ring $\widetilde{\mathcal{R}} = \mathcal{R}[y]/\langle y^{\widetilde{r_0}} - 1\rangle$. An example of BQC parity-check matrix with $n_0 = 3$, $r_0 = 1$, $\widetilde{r_0} = 2$ is the following:

$$H = \left[\mathsf{a}(y) \;\middle|\; \mathsf{b}(y) \;\middle|\; \mathsf{c}(y)\right] = \left[\begin{array}{cc|cc|cc} \mathsf{a}_0(x) & \mathsf{a}_1(x) & \mathsf{b}_0(x) & \mathsf{b}_1(x) & \mathsf{c}_0(x) & \mathsf{c}_1(x) \\ \mathsf{a}_1(x) & \mathsf{a}_0(x) & \mathsf{b}_1(x) & \mathsf{b}_0(x) & \mathsf{c}_1(x) & \mathsf{c}_0(x) \end{array}\right]$$

Since $n = n_0 \cdot (\widetilde{r_0}p)$, a BQC code with same rate and (almost) same length as a QC code can be obtain via a $\widetilde{r_0}$-times reduction in p. Moreover, it is easy to show that computing the inverse of a element of $\widetilde{\mathcal{R}}$ only requires the inverse of a *single* element of $\mathcal{R}$. For example, the inverse of a polynomial $\mathsf{a}(y) \in \widetilde{\mathcal{R}}$ when $\widetilde{r_0} = 2$ (i.e., $\mathsf{a}(y) = \mathsf{a}_0(x) + y\mathsf{a}_1(x)$) is $\mathsf{a}(y)^{-1} = (\mathsf{a}_0(x) + y\mathsf{a}_1(x))\left(\mathsf{a}_0(x)^2 + \mathsf{a}_1(x)^2\right)^{-1}$. We can thus compute the inverse of an element of $\widetilde{\mathcal{R}}$ at the cost of inverting a single polynomial in $\mathcal{R}$, which is $\widetilde{r_0}$-times smaller than the original object.

Let the Hamming weight of $\mathsf{a}(y)=[\mathsf{a}_0(x), \ldots, \mathsf{a}_{\widetilde{r_0}-1}(x)] \in \widetilde{\mathcal{R}}$ be $\mathtt{wt}(\mathsf{a}(y))=\sum_{i=0}^{\widetilde{r_0}-1} \mathtt{wt}(\mathsf{a}_i(x))$. We can now define the family of BQC-MDPC codes as the set of BQC codes with $r_0 = 1$ and rate $\frac{n_0-1}{n_0}$ that admit a sparse representation, i.e., where the blocks $\mathsf{h}_0(y), \ldots, \mathsf{h}_{n_0-1}(y)$ have Hamming weight $v \in \mathcal{O}(\sqrt{n})$. BQC-MDPC codes are (v, vn_0)-regular and can thus be successfully decoded by bit flipping algorithms. Note that, while for QC-MDPC codes we can set the Hamming weight of all the polynomials in $\mathcal{R}$ to a single (odd) value, in order to guarantee the existence of the systematic form of H we must choose different Hamming weights $(v_0, \ldots, v_{\widetilde{r_0}-1})$ for the polynomials in $\mathcal{R}$ composing each block.

In SLEEK-E, we employ BQC-MDPC codes with $n_0 = 2$ and $\widetilde{r_0} = 3$ (thus, with length $6p$ and rate $\frac{1}{2}$). Choosing $n_0 = 2$ allows us to minimize the combined size of public key and ciphertext: differently from the case of long-term keys, where the ciphertext size can be reduced by increasing n_0 and thus public key size, for ephemeral keys this tradeoff is not beneficial, as the public key has to be transmitted each time. We employ $\widetilde{r_0} = 3$ as it provides the best combination of circulant block size p and arithmetic complexity for operations over $\widetilde{\mathcal{R}}$. For such parameters, any element of $\widetilde{\mathcal{R}}$ is invertible if and only if $v_0 + v_1 + v_2$ is odd; we report the inverse computation procedure showing this fact in Section 4.3. Without loss of generality, we set v_0 to be odd, and $v_1 = v_2$ to be even so that either $v_1 = v_2 = v_0+1$ or $v_1 = v_2 = v_0-1$, in order to maintain the weight of the three blocks as balanced as possible, and thus maximize the hardness of key recovery.

4.2 Scheme Design and Security

BQC-MDPC Niederreiter PKE. Instantiating the Niederreiter template with BQC-MDPC codes yields essentially the same PKE shown in Figure 1. The OW-CPA security of the resulting PKE, i.e., the hardness of message and key recovery attacks against it, can be reduced to the hardness of BQC-SDP and BQC-CFP exactly as for QC codes, where

the BQC- prefix indicates the usage of BQC codes instead of QC. We now analyse both the theoretical and concrete hardness of solving such problems. First, it is easy to show that BQC-SDP and BQC-CFP are both NP-hard problems, under the same assumptions underlying the NP-hardness of QC-SDP and QC-CFP.

Lemma 2. *BQC-SDP is as hard as QC-SDP, and BQC-CFP is as hard as QC-CFP.*

Proof. We show that BQC-SDP is as hard as QC-SDP by providing a reduction from the latter to the former; the same reduction applies to the case of CFP. Given an instance of QC-SDP, i.e., a $r_0 p \times n_0 p$ parity-check matrix H, a syndrome $\mathbf{s} \in \mathbb{F}_2^{r_0 p}$, and an integer t, one can construct a $\widetilde{r_0} \cdot r_0 p \times \widetilde{r_0} \cdot n_0 p$ BQC parity-check matrix H' (for any arbitrary $\widetilde{r_0}$) by substituting each $p \times p$ circulant block in H with a $\widetilde{r_0} p \times \widetilde{r_0} p$ block matrix filled with zeros, save for $\widetilde{r_0}$ copies of the original circulant block on the main diagonal. The corresponding syndrome $\mathbf{s}'$ is obtained by replicating $\widetilde{r_0}$ times each p-sized block composing $\mathbf{s}$. The original QC-SDP instance has a valid solution (i.e., there is an error vector $\mathbf{e}$ with Hamming weight $\leq t$ such that $\mathbf{s} = H\mathbf{e}^\intercal$) if and only if the new BQC instance has a valid solution with weight $\leq \widetilde{r_0} t$. □

The *concrete* hardness of BQC-SDP and BQC-CFP is, however, clearly different from the one of QC-SDP and QC-CFP, respectively. We now show how the additional structure of BQC codes influences the hardness of message and key recovery attacks.

Message Recovery Attack. Given a SLEEK-E public key and ciphertext, i.e., a BQC-MDPC matrix M in systematic form and a syndrome $\mathbf{s}$ in the form

$$M = [\mathsf{m}_0(y) \mid 1] = \left[\begin{array}{ccc|ccc} \mathsf{m}_{0,0}(x) & \mathsf{m}_{0,1}(x) & \mathsf{m}_{0,2}(x) & 1 & 0 & 0 \\ \mathsf{m}_{0,2}(x) & \mathsf{m}_{0,0}(x) & \mathsf{m}_{0,1}(x) & 0 & 1 & 0 \\ \mathsf{m}_{0,1}(x) & \mathsf{m}_{0,2}(x) & \mathsf{m}_{0,0}(x) & 0 & 0 & 1 \end{array}\right] \quad \mathbf{s} = [\mathsf{s}(y)] = \begin{bmatrix} \mathsf{s}_0(x) \\ \mathsf{s}_1(x) \\ \mathsf{s}_2(x) \end{bmatrix},$$

one has to find an error vector $\mathbf{e}=[\mathsf{e}_0(y) \,|\, \mathsf{e}_1(y)]=[\mathsf{e}_{0,0}(x)\ \mathsf{e}_{0,1}(x)\ \mathsf{e}_{0,2}(x) \,|\, \mathsf{e}_{1,0}(x)\ \mathsf{e}_{1,1}(x)\ \mathsf{e}_{1,2}(x)]$ such that $\mathtt{wt}(\mathbf{e}) = t$ and $\mathbf{s} = H\mathbf{e}^\intercal$. This BQC-SDP instance can be solved via ISD techniques, with a DOOM speedup equal to $\widetilde{r_0} p$ ($= 3p$ in our case), requiring $\mathtt{ISD}(6p, 3p, t)/\sqrt{3p}$ bit operations. The attack complexity can be reduced by a polynomial (in p) factor by evaluating the public polynomial at $y = 1$ and considering the following QC-SDP instance on a smaller QC code with length $n_0 p$ ($= 2p$):

$$M' = [\mathsf{m}_{0,0}(x) + \mathsf{m}_{0,1}(x) + \mathsf{m}_{0,2}(x) \mid 1] \quad \mathbf{s}' = [\mathsf{s}_0(x) + \mathsf{s}_1(x) + \mathsf{s}_2(x)]$$

admitting as solution the vector $\mathbf{e}' = [\mathsf{e}_{0,0}(x) + \mathsf{e}_{0,1}(x) + \mathsf{e}_{0,2}(x) \mid \mathsf{e}_{1,0}(x) + \mathsf{e}_{1,1}(x) + \mathsf{e}_{1,2}(x)]$ with Hamming weight $\leq t$. As already shown in Lemma 1, once $\mathbf{e}'$ is found, it is easy to recover the original PKE message $\mathbf{e}$. This formulations has two fundamental advantages. First, the size of the objects at hand is $3\times$ smaller, improving the cost of each ISD iteration. Second, the vector $\mathbf{e}'$ does not have a fixed Hamming weight, since the block-wise sums can lead to self-cancellation in the polynomial coefficients: if $\mathbf{e}'$ has a weight $t' < t$, then the number of ISD iterations required to find it is lower. To quantify the security reduction given by this attack, we analyse the probability distribution of $\mathtt{wt}(\mathbf{e}')$.

Proposition 1. *Let $\mathcal{T}$ be the random variable corresponding to the Hamming weight of $\mathbf{e}'$. Then, for every integer $0 \leq x \leq t$ s.t. $t - x$ is even:*

$$\Pr(\mathcal{T} = x) = \frac{\binom{2p}{x}}{\binom{6p}{t}} \sum_{i=0}^{\min\{\frac{t-x}{2}, x\}} \binom{x}{i} \binom{2p - x}{\frac{t-x}{2} - i} 3^{\frac{t+x}{2} - 2i}$$

Proof. First, we have that the total number of binary vectors with length $2p$ nd weight x is $\binom{2p}{x}$, while the total number of arrangements of t ones within the original error vector is

$\binom{6p}{t}$. We then derive our formulation of $\Pr(\mathcal{T}=x)$ via a counting argument, computing the total number of arrangements that lead to $\mathcal{T}=x$. Consider $\mathbf{e}''$ as $\mathbf{e}'$ computed over the integers, $\mathbf{e}''=[\varphi(\mathsf{e}_{0,0}(x))+\varphi(\mathsf{e}_{0,1}(x))+\varphi(\mathsf{e}_{0,2}(x))\,|\,\varphi(\mathsf{e}_{1,0}(x))+\varphi(\mathsf{e}_{1,1}(x))+\varphi(\mathsf{e}_{1,2}(x))]$, where φ is a map between $\mathbb{F}_2^p$ and $\mathbb{Z}^p$. The entries of $\mathbf{e}''$ are in $\{0,1,2,3\}$. Let i_{val} be the number of coordinates where ${\mathbf{e}''}_j = val$. We have that $i_1+i_3=x$ and $i_1+2i_2+3i_3=t$, from which we get $i_1=x-i_3$ and $i_2=\frac{t-x}{2}-i_3$. The positions where ${\mathbf{e}''}_j=3$ can be selected in $\binom{x}{i_3}$ ways, while the positions where ${\mathbf{e}''}_j=2$ can be selected in $\binom{2p-x}{\frac{t-x}{2}-i_3}$ ways (the positions where ${\mathbf{e}''}_j=0$ and ${\mathbf{e}''}_j=1$ are then fixed). Finally, for each of the $x-i_3$ coordinates where ${\mathbf{e}''}_j=1$, we can select which polynomial among $\{\mathsf{e}_{*,0}(x),\mathsf{e}_{*,1}(x),\mathsf{e}_{*,2}(x)\}$ has $\mathbf{e}_{*,h_j}=1$ in $\binom{3}{1}=3$ ways, while for each of the $\frac{t-x}{2}-i_3$ coordinates where ${\mathbf{e}''}_j=2$, we can select which polynomial among $\{\mathsf{e}_{*,0}(x),\mathsf{e}_{*,1}(x),\mathsf{e}_{*,2}(x)\}$ has $\mathbf{e}_{*,h_j}=1$ in $\binom{3}{2}=3$ ways. The formula for $\Pr(\mathcal{T}=x)$ can be found iterating over all possible values of i_3. □

By taking the variable Hamming weight of $\mathbf{e}'$ into account, we can compute the average complexity of solving the SDP instance, considering also the fact that an attacker with any amount of ciphertexts can tune its effort to solve only the easiest instances, i.e., the ones with a Hamming weight $\leq t^*$ (for the optimal choice of t^*):

$$\min_{t^*}\left\{\frac{1}{\Pr(\mathcal{T}\leq t^*)}\cdot\sum_{\substack{x=0\\ t-x\ \text{even}}}^{t^*}\Pr(\mathcal{T}=x\mid\mathcal{T}\leq t^*)\cdot\frac{\mathtt{ISD}(6p,3p,x)}{\sqrt{p}}\right\}$$

Key Recovery Attack. We now summarize the results for the key recovery attack, that follow the same line of reasoning of the message recovery attack. Given a SLEEK-E public key M, one has to recover one of the rows of its sparse representation $H=[\mathsf{h}_0(y)\,|\,\mathsf{h}_1(y)]$, where $\mathsf{h}_0(y)=\mathsf{h}_{0,0}(x)+y\mathsf{h}_{0,1}(x)+y^2\mathsf{h}_{0,2}(x)$ and $\mathsf{h}_1(y)=\mathsf{h}_{1,0}(x)+y\mathsf{h}_{1,1}(x)+y^2\mathsf{h}_{1,2}(x)$. Solving such CFP instance via ISD yields a computational complexity of $\mathtt{ISD}(6p,3p,w)/3p$, where $w=2v=2(v_0+v_1+v_2)$. If we evaluate the public polynomial at $y=1$, we get a reduced instance with public matrix M', admitting as sparse dual codeword the vector $\mathbf{h}'=[\mathsf{h}_{0,0}(x)+\mathsf{h}_{0,1}(x)+\mathsf{h}_{0,2}(x)\,|\,\mathsf{h}_{1,0}(x)+\mathsf{h}_{1,1}(x)+\mathsf{h}_{1,2}(x)]$. Again, the Hamming weight of $\mathbf{h}'$ is $\leq w$ and not fixed.

Proposition 2. *Let $\mathcal{W}$ be the random variable corresponding to the Hamming weight of $\mathbf{h}'$. Then, for every integer $0\leq x\leq w$ s.t. $w-x$ is even:*

$$\Pr(\mathcal{W}=x)=\sum_{\substack{i=\max\{0,x-v\}\\ v-i\ even}}^{\min\{v,x\}}\Phi(i)\Phi(x-i)$$

$$\Phi(i)=\frac{1}{\binom{p}{v_1}\binom{p}{v_2}}\sum_{\substack{j=\max\{0,i-v_2\}\\ v_0+v_1-j\ even}}^{\min\{v_0+v_1,i\}}\binom{v_0}{(v_0+v_1-j)/2}\binom{p-v_0}{v_1+(j+v_1-v_0)/2}\binom{j}{(j+v_2-i)/2}\binom{p-j}{v_2+(i+v_2-j)/2}$$

Proof. We model the Hamming weight of $\mathbf{h}'$ considering its two halves separately, and analysing one addition at a time. The probability that $\mathtt{wt}(\mathsf{h}_{0,0}(x)+\mathsf{h}_{0,1}(x))=j$ is $\binom{v_0}{(v_0+v_1-j)/2}\binom{p-v_0}{v_1+(j+v_1-v_0)/2}/\binom{p}{v_1}$. Conditioned over this event, the probability that $\mathtt{wt}(\mathsf{h}_{0,0}(x)+\mathsf{h}_{0,1}(x)+\mathsf{h}_{0,1}(x))=i$ is $\binom{j}{(j+v_2-i)/2}\binom{p-j}{v_2+(i+v_2-j)/2}/\binom{p}{v_2}$. Iterating over all possible values of i and j, we obtain the stated formula. □

By taking the variable Hamming weight of $\mathbf{h}'$ into account, we can compute the average complexity of solving the KRA instance as:

$$\min_{w^*}\left\{\frac{1}{\Pr(\mathcal{W}\leq w^*)}\cdot\sum_{\substack{x=0\\ w-x\ \text{even}}}^{w^*}\Pr(\mathcal{W}=x\mid\mathcal{W}\leq w^*)\cdot\frac{\mathtt{ISD}(6p,3p,x)}{p}\right\},$$

KEM construction. By selecting appropriate parameters for the employed BQC-MDPC codes, we get an OW-CPA secure Niederreiter PKE, that can be used alongside the

construction shown in Figure 2 to obtain an IND-CCA KEM. While just achieving IND-CPA guarantees would allow to strip a considerable amount of the FO transform from SLEEK, we choose to keep both the verifiability of the pseudorandom generation of the error and the ward against multikey attacks. We can nonetheless improve the performance of SLEEK-E in the following ways. First, we move from a $\mathsf{U}^{\perp}$ to a $\mathsf{U}^{\perp}_{m}$ transform, thus saving the time from the hashing of the ciphertext (Line 8 in Encaps and Line 7 in Decaps). Then, we target DFR values significantly higher than $2^{-\lambda}$, as long as they do not impact system reliability (i.e., 2^{-32} and 2^{-64}). In doing so, the code length of the BQC-MDPC code we employ is expected to shorten.

4.3 Efficient Implementation

In this section, we discuss our efficient implementation of the arithmetic operations over $\widetilde{\mathcal{R}}$ required in the BQC-MDPC based Niederreiter PKE, tailored for an IND-CPA ephemeral key scenario. When ephemeral keys are required, e.g., to guarantee forward secrecy, users usually fall within one of two common scenarios. In the first setting, an user generates a single keypair, and employs it a small amount of times to avoid the repeated overhead of key generation. Thanks to SLEEK-E KEM construction, this usage is still (moderately) secure despite the higher DFR levels, as the verifiability of the pseudorandom generation of the PKE message, along with the nonetheless low DFR target ($\leq 2^{-32}$), provide a margin of resistance against decoding failures-based attacks. In the second setting, users correctly generate a fresh keypair each time. When this is done, an active attacker can only collect a single timing sample, meaning that we can discuss how to further optimize the cryptosystem implementation by relaxing constant-time execution constraints. Thanks to the efficient key generation provided by the usage of BQC-MDPC codes, the timing improvement due to this last optimization is *larger* than the timing cost of key generation, rendering the generation of truly ephemeral keypairs advantageous.

Polynomial Multiplication over $\widetilde{\mathcal{R}}$. To perform multiplication between two elements $\mathsf{a}(y), \mathsf{b}(y) \in \widetilde{\mathcal{R}}$ we implement the evaluation-interpolation algorithm based on the Chinese Remainder Theorem (CRT). The ring modulus (y^3-1) factors into two coprime polynomials, $y-1$ and y^2+y+1, therefore $\widetilde{\mathcal{R}}$ decomposes into the direct product $\mathcal{R}[y]/\langle y^3-1\rangle \cong \mathcal{R}[y]/\langle y-1\rangle \times \mathcal{R}[y]/\langle y^2+y+1\rangle$. The projection of $\mathsf{a}(y) = \mathsf{a}_0(y) + y\mathsf{a}_1(y) + y^2\mathsf{a}_2(y)$ (the same holds for $\mathsf{b}(y)$) $\mathcal{R}[y]/\langle y-1\rangle$ is $\mathsf{a}'(x) = \mathsf{a}_0(x) + \mathsf{a}_1(x) + \mathsf{a}_2(x)$, while its projection over $\mathcal{R}[y]/\langle y^2+y+1\rangle$ is $\mathsf{a}''(y) = (\mathsf{a}_0(x)+\mathsf{a}_1(x))+y(\mathsf{a}_0(x)+\mathsf{a}_2(x))$. The product $\mathsf{c}_0(x) = \mathsf{a}'(x)\mathsf{b}'(x)$ requires a single polynomial multiplication over $\mathcal{R}$, while the product $\mathsf{a}''(y)\mathsf{b}''(y)$ can be computed via Karatsuba algorithm, requiring 3 multiplications. The final result can then be found by combining the two result. This strategy requires 4 modular multiplications and 12 modular additions.

Systematic Form Computation. To compute the systematic form of H, $\mathsf{m}_0(y) = \mathsf{h}_0(y) \cdot \mathsf{h}_1(y)^{-1}$, we merge the inverse computation and the product in a single step. We project the operands over the rings $\mathcal{R}[y]/\langle y-1\rangle$ and $\mathcal{R}[y]/\langle y^2+y+1\rangle$, where inverting $\mathsf{h}_1(y)$ requires finding the inverse of the following polynomials:

$$\begin{cases} \mathsf{d}_0(x) = \mathsf{h}_{1,0}(x) + \mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x) \\ \mathsf{d}_1(x) = (\mathsf{h}_{1,0}(x)+\mathsf{h}_{1,2}(x))(\mathsf{h}_{1,1}(x)+\mathsf{h}_{1,2}(x)) + (\mathsf{h}_{1,0}(x)+\mathsf{h}_{1,2}(x))^2 + (\mathsf{h}_{1,1}(x)+\mathsf{h}_{1,2}(x))^2 \end{cases}$$

To find $\mathsf{d}_0(x)^{-1}$ and $\mathsf{d}_1(x)^{-1}$, we need a single polynomial inversion $\mathsf{inv}(x) = (\mathsf{d}_0(x)\mathsf{d}_1(x))^{-1}$, since then $\mathsf{d}_0(x)^{-1} = \mathsf{inv}(x)\mathsf{d}_1(x)$ and $\mathsf{d}_1(x)^{-1} = \mathsf{inv}(x)\mathsf{d}_0(x)$. The complete procedure requires a total of 15 additions, 10 multiplications (highlighted in red), 2 squarings, and one polynomial inversion (highlighted in blue) over $\mathcal{R}$.

$$\begin{cases}
\mathsf{d}_0(x) = \mathsf{h}_{1,0}(x) + \mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x) \\
\mathsf{d}_1(x) = (\mathsf{h}_{1,0}(x) + \mathsf{h}_{1,2}(x)) \times (\mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x)) + (\mathsf{h}_{1,0}(x) + \mathsf{h}_{1,2}(x))^2 + (\mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x))^2 \\
\mathsf{inv}(x) = (\mathsf{d}_0(x) \times \mathsf{d}_1(x))^{-1} \\
\mathsf{d}_1(x)^{-1} = \mathsf{inv}(x) \times \mathsf{d}_0(x) \\
\mathsf{k}_0(x) = \mathsf{inv}(x) \times \mathsf{d}_1(x) \\
\mathsf{k}_1(x) = (\mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x)) \times \mathsf{d}_1(x)^{-1} \\
\mathsf{k}_2(x) = (\mathsf{h}_{1,0}(x) + \mathsf{h}_{1,2}(x)) \times \mathsf{d}_1(x)^{-1} \\
\mathsf{p}_0(x) = (\mathsf{h}_{0,0}(x) + \mathsf{h}_{0,2}(x)) \times (\mathsf{k}_1(x) + \mathsf{k}_2(x)) \\
\mathsf{p}_1(x) = (\mathsf{h}_{0,1}(x) + \mathsf{h}_{0,2}(x)) \times \mathsf{k}_1(x) \\
\mathsf{p}_2(x) = (\mathsf{h}_{0,0}(x) + \mathsf{h}_{0,1}(x)) \times \mathsf{k}_2(x) \\
\mathsf{v}_0(x) = (\mathsf{h}_{0,0}(x) + \mathsf{h}_{0,1}(x) + \mathsf{h}_{0,2}(x)) \times \mathsf{k}_0(x) \\
\mathsf{v}_1(x) = \mathsf{p}_0(x) + \mathsf{p}_1(x) \\
\mathsf{v}_2(x) = \mathsf{p}_0(x) + \mathsf{p}_2(x) \\
\mathsf{m}_{0,0}(x) = \mathsf{v}_0(x) + \mathsf{v}_2(x) \\
\mathsf{m}_{0,1}(x) = \mathsf{v}_0(x) + \mathsf{v}_1(x) \\
\mathsf{m}_{0,2}(x) = \mathsf{v}_0(x) + \mathsf{v}_1(x) + \mathsf{v}_2(x)
\end{cases}$$

BQC-MDPC based SLEEKDEC **Decoder.** The specification of SLEEKDEC algorithms remains essentially unchanged when instantiated with BQC-MDPC codes. Each p-sized block of the `upc` vector is computed as the sum over $\mathbb{Z}$ of $v_0 + v_1 + v_2$ rotations of the three polynomials composing the syndrome. The term $(H_{n_0-1}^{-1}\mathbf{s})$ required during the last decoding iteration can be efficiently computed by storing in the secret key the following terms, computed during the key generation phase:

$$\begin{cases}
\mathsf{k}_0(x) = \mathsf{d}_0(x)^{-1} \\
\mathsf{k}_1(x) = (\mathsf{h}_{1,1}(x) + \mathsf{h}_{1,2}(x))\mathsf{d}_1(x)^{-1} \\
\mathsf{k}_2(x) = (\mathsf{h}_{1,0}(x) + \mathsf{h}_{1,2}(x))\mathsf{d}_1(x)^{-1}
\end{cases}$$

The procedure requires 4 multiplications (highlighted in red) and 10 additions over $\mathcal{R}$, without the need to explicitly compute $H_{n_0-1}^{-1}$:

$$\begin{cases}
\mathsf{p}_0(x) = (\mathsf{s}_0(x) + \mathsf{s}_2(x)) \times (\mathsf{k}_1(x) + \mathsf{k}_2(x)) \\
\mathsf{p}_1(x) = (\mathsf{s}_1(x) + \mathsf{s}_2(x)) \times \mathsf{k}_1(x) \\
\mathsf{p}_2(x) = (\mathsf{s}_0(x) + \mathsf{s}_1(x)) \times \mathsf{k}_2(x) \\
\mathsf{v}_0(x) = (\mathsf{s}_0(x) + \mathsf{s}_1(x) + \mathsf{s}_2(x)) \times \mathsf{k}_0(x) \\
\mathsf{v}_1(x) = \mathsf{p}_0(x) + \mathsf{p}_1(x) \\
\mathsf{v}_2(x) = \mathsf{p}_0(x) + \mathsf{p}_2(x) \\
\mathsf{e}_0(x) = \mathsf{v}_0(x) + \mathsf{v}_2(x) \\
\mathsf{e}_1(x) = \mathsf{v}_0(x) + \mathsf{v}_1(x) \\
\mathsf{e}_2(x) = \mathsf{v}_0(x) + \mathsf{v}_1(x) + \mathsf{v}_2(x)
\end{cases}$$

Finally, to reduce the decoder runtime in settings where constant-time execution in not strictly required, we replace the `rotateBig` part of the circulant shift algorithm with simple pointer arithmetic, and we stop the decoder after the *second* iteration if the iterative procedure has already converged.

5 Experimental Results

In this section, we present the resulting parameter sets, benchmark the proposed decoder, and compare our schemes against state-of-the-art implementations of BIKE and HQC. Benchmarks were performed on an Intel Core i7-12700K at 3.8 GHz with Turbo Boost disabled, using `rdtscp` for cycle counting. All IND-CCA implementations are constant-time under the standard assumption that bitwise operations and carry-less multiplications execute in constant time.

5.1 Design Space Exploration Criteria

Given the PKE structure and constructions of SLEEK, three free parameters can be tweaked to obtain different SLEEK-LT instantiations: *1)* the security margin λ quantifying the cost of the most efficient attack as the effort of breaking AES with a λ-bit key, *2)* the number of circulant blocks n_0 of the matrix H, which defines the code rate $(n_0-1)/n_0$, and *3)* the decoding failure rate of the QC-MDPC being employed, which, for the sake of clarity, we will express as $\mathsf{dfr} := -\log_2(\text{DFR})$ (where $\text{DFR} = 2^{-a} \leq 2^{-\lambda}$). The free parameters that can be tweaked for SLEEK-E are: *1)* the security margin λ and *2)* the decoding failure rate of the BQC-MDPC code. In SLEEK-E, we maintain $n_0 = 2$ and $\widetilde{r_0} = 3$ as they provide the best performance figures. We consider bit security levels matching NIST requirements ($\lambda \in \{128, 192, 256\}$), plus an additional $\lambda = 80$ security margin, matching the one of lightweight ciphers such as the ISO Standard cipher Present [BKL+07, ISO19], which was shown to take a considerable amount of quantum resources to be broken [RP21], and was also considered for the lightweight cipher context winner ASCON [DEMS21, TMC+25].

For SLEEK-LT, the λ and dfr should make the inequality $\mathsf{dfr} \geq \lambda$ hold, due to the requirement on the δ-correctness of the underlying PKE. Since decreasing the DFR of the involved QC-MDPC codes implies additional computational and storage costs, we consider only the plane $\mathsf{dfr} = \lambda$ in the parameter space, while freely varying n_0. We also consider 2^{-160} for parameter sets where $\lambda > 160$: this choice was adopted in the standardization of ML-KEM by NIST [Nat24a] to improve the performance of Kyber, while retaining IND-CCA guarantees in practice. The justification of this claim is that the 2^{160} calls to the decryption oracle which the attacker needs to perform are unlikely to be feasible, as an honest decryption oracle will not provide so many answers, ever. Orthogonally to the λ and dfr tuning, choosing a higher rate code (i.e., one with a higher value for n_0) yields smaller ciphertexts at the cost of a larger public key and decoding time. Figure 3 reports a graphical depiction of the design space we analyzed, and a comparison with the current state of the art, which targets IND-CCA security. The figure illustrates the parameter choices for previous designs following the same Niederreiter template as SLEEK, BIKE [ABB+23] and LEDAcrypt-KEM [BBC+20], along with the NIST selected HQC [GMA+25],. All the previous designs target only $\lambda \in \{128, 192, 256\}$, and, while BIKE focuses only on $n_0 = 2$, LEDAcrypt-KEM proposes parameter sets for $n_0 \in \{2, 3, 4\}$. Considering HQC, the NIST selected standard for code based KEMs, we note that, since it is built on Alekhnovich's template instead of Niederreiter's, it does not enjoy the flexibility of changing the code rate from 1/2. In SLEEK our exploration considers values of n_0 between 2 and 8, evaluating a broader sets of trade-offs coming from changing the code rate. We note that such a change does not

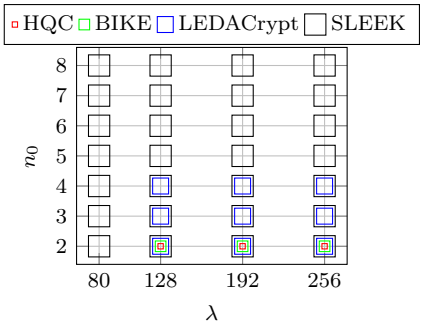


Figure 3: Design space for IND-CCA KEMs, expressed as a function of the design parameters λ and n_0.

Table 1: Code parameters and computational hardness of SLEEK-LT (short list).

λ-$\mathbf{n_0}$-dfr **Target**	p	v	t	th_1	th_2	th_3	**Actual** dfr	λ_{cm}	λ_{ck}	λ_{qm}	λ_{qk}
80-2-80	7019	47	87	29	24	24	80.32	95.23	96.50	128.04	128.49
80-5-80	4091	57	38	34	32	29	83.46	96.03	95.74	126.24	128.90
80-8-80	3533	59	30	35	34	30	81.66	97.56	95.39	126.78	130.46
128-2-128	14341	71	136	42	37	36	128.46	143.54	143.34	179.63	178.64
128-5-128	8363	89	59	52	48	45	128.07	144.09	146.07	178.01	182.41
128-8-128	7013	91	46	53	50	46	128.02	144.82	143.56	177.64	181.69

Table 2: Code parameters and computational hardness of SLEEK-E.

λ-dfr **Target**	p	(v_0, v_1, v_2)	t	th_1	th_2	th_3	**Actual** dfr	λ_{cm}	λ_{ck}	λ_{qm}	λ_{qk}
80-32	2069	(15, 16, 16)	88	29	25	25	33.02	95.03	95.03	122.97	123.46
80-64	2269	(15, 16, 16)	88	29	25	24	64.41	95.15	95.16	123.35	123.76
128-32	4219	(25, 24, 24)	138	44	39	37	33.83	143.92	145.40	175.78	177.79
128-64	4483	(25, 24, 24)	137	44	38	37	66.55	143.04	145.47	175.01	177.97
192-32	7949	(35, 36, 36)	202	63	57	54	32.73	207.04	211.84	242.29	247.88
192-64	8363	(35, 36, 36)	202	63	56	54	64.90	207.14	211.99	242.47	248.01
256-32	12491	(45, 46, 46)	267	80	73	69	33.95	271.31	271.00	309.10	309.33
256-64	13037	(45, 46, 46)	267	80	72	69	65.52	271.40	271.09	309.24	309.44

weaken the computational hardness of the underlying problem [ABP26].

For SLEEK-E, we are allowed to relax the requirement on the DFR as the attacker has only a single decryption oracle query before the key is discarded. In particular, we consider two DFR levels, namely 2^{-32} and 2^{-64}, chosen so that the failure events are not just below the common reliability targets for functional systems (somewhere between 10^{-5} and 10^{-7}) but low enough to be considered acceptable or even negligible in tight reliability scenarios, e.g., the highest Security Integrity Level (SIL) in [ISO10] (i.e., SIL 4) calls for a failure probability of 10^{-9} in an hour of device functioning. Finally, a further argument for target DFRs such as 2^{-32} and 2^{-64} is the one of providing some degree of protection against accidental reuses of ephemeral keys due to either bugs, or improper inclusion in protocols which would need stronger guarantees. Indeed, our IND-CPA KEM structure is intentionally kept close to the IND-CCA one so that, unless an actual decryption failure takes place, a modicum of key reuse is possible without losing guarantees.

5.2 Parameters for SLEEK

In this section, we report the parameter sets obtained, given λ, n_0, and a target DFR, by selecting the smallest values of p, t which make amount of Boolean operations to perform a Message Recovery Attack (MRA) $2^{\lambda_{cm}}$ as expensive as breaking AES with a λ bit key on a classical computer ($\approx 2^{\lambda+15}$ [BBC+20]), and then select the smallest value for v which makes the Key Recovery attacks take at least the same computational effort, taking $2^{\lambda_{ck}}$. Analogous estimates are derived for attacks performed with a quantum computer, for which we denote as $2^{\lambda_{qm}}$ and $2^{\lambda_{qk}}$ the number of quantum gates taken for the message and key recover attacks, respectively. These criteria match the one employed by NIST to define the security margin of the proposals to its standardization effort, therefore allowing us a direct comparison of our designs with BIKE and HQC. The computational effort was estimated by means of three complexity estimation tools, Cryptographic Estimators [Cry26], LEDAtools [PBL25] and the one of the recent MAGE-Stern attack [CHLT26], retaining for each parameter set the smallest between the three estimates. To guarantee a conservative estimate of the security of SLEEK against static attacks, we considered a logarithmic memory access cost. We note that, under such security estimate criterion, all parameter

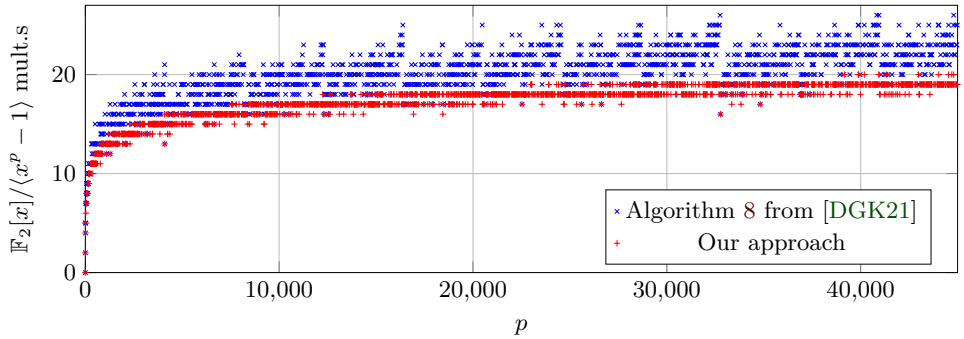


Figure 4: No. of polynomial multiplications required to compute a polynomial inverse as a function of the bit size of the operands (i.e., p).

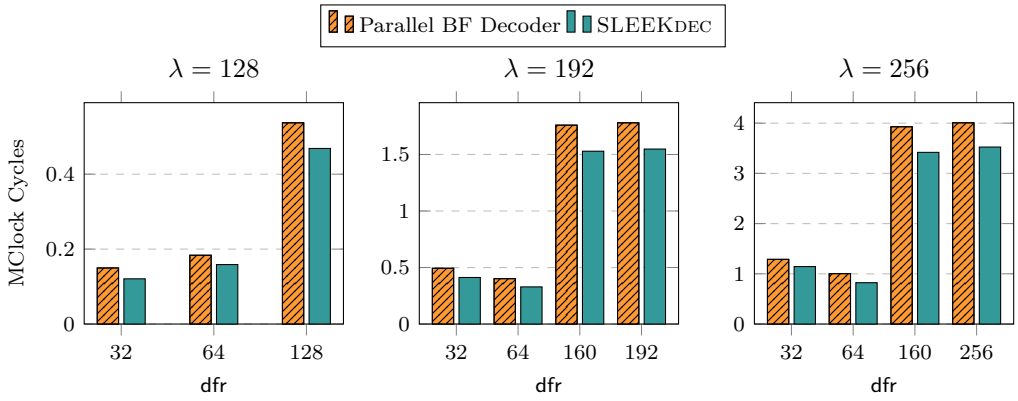


Figure 5: Clock cycles required by the parallel BF and SLEEK decoders ($n_0 = 2$).

sets of BIKE fall short of the required security level [CHLT26]. The DFR estimates are performed according to the extension to three decoder iterations [ABPP26] of the method proposed in [ABP24]. For the sake of brevity, we denote as dfr the value $\mathsf{dfr} = -\log_2(\text{DFR})$.

Table 1 reports a subset of the obtained parameters with $n_0 \in \{2, 5, 8\}$ for SLEEK-LT (with full results in Appendix A). The reported figures show how our design framework is able to provide a tight match for the required security margin and DFR, thanks to the closeness between consecutive admissible values of p. Moving from the lowest to the highest code rate (i.e., from the lowest to the highest value of n_0) reduces by more than 50% the length p of the binary polynomials we employ, across all security levels and DFRs. Table 2 shows the parameters for SLEEK-E. As it can be seen, targeting a DFR level higher than 2^λ but still negligible from a practical standpoint allows for a reduction in size of the objects manipulated by the cipher. The exploration campaign also highlights the wide range of polynomial lengths with which our arithmetic primitives will work, which is between 2069 (parameter set for SLEEK-E 80-32) and 44531 (parameter set for SLEEK-LT 256-2-256).

5.3 Evaluation of Polynomial Inversion and SLEEKDEC

Polynomial inversion. To quantify the gain of our proposed approach for polynomial inversion, Figure 4 shows the number of multiplications required by Alg. 8 (blue) and by our approach (red) to compute $\mathsf{h}(x)^{-1} = \mathsf{h}(x)^{2^{p-1}-2}$, as a function of the value of p. Our strategy provides on average a 10% to 20% reduction in the number of required multiplications with respect to Alg. 8, and thanks to the relatively smooth behavior of the multiplication counting function of our addition chains, choosing the cryptosystem

Table 3: Performance comparison between SLEEK-LT, BIKE and HQC. KG stands for key generation, ENC for encapsulation, DEC for decapsulation.

Scheme	λ-$\mathbf{n_0}$-dfr	sk	pk	ctx	ptx	KG	ENC	DEC	ENC+ DEC
			(B)				(kCC)		
SLEEK	80-2-80	937	878	902	32	167	40	204	245
	80-5-80	569	2048	536	32	105	31	333	364
	80-8-80	505	3094	466	32	120	33	435	468
SLEEK	128-2-128	1857	1793	1817	32	451	88	573	662
	128-5-128	1105	4184	1070	32	445	100	1217	1318
	128-8-128	937	6139	901	32	276	77	1353	1430
BIKE	128-2-128	5223	1541	1573	32	451	138	1684	1822
HQC	128	2321	2241	4433	32	69	138	312	450

Table 4: Performance comparison between SLEEK-E, BIKE, Lean-BIKE, and HQC.

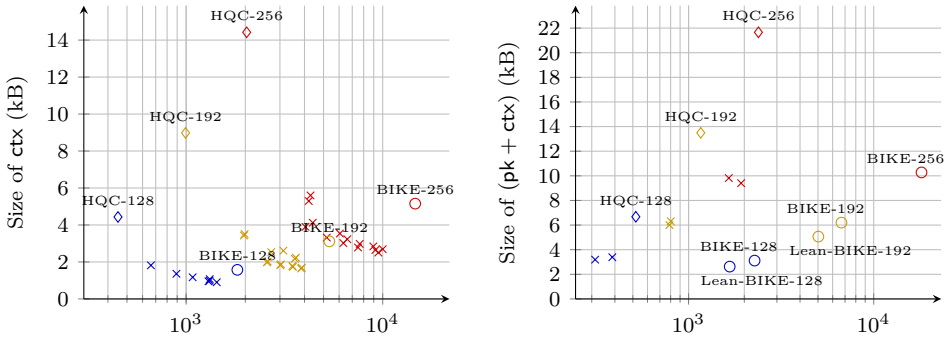
Scheme	λ-dfr	sk	pk	ctx	ptx	KG	ENC	DEC	KEYGEN+ +ENC+DEC	pk+ ctx
			(B)				(kCC)			
SLEEK	80-32	849	777	801	32	53	19	65	137	1578
	80-64	921	852	876	32	60	19	68	147	1728
	128-32	1641	1584	1608	32	115	33	163	312	3192
	128-64	1761	1683	1707	32	156	38	192	387	3390
	192-32	3081	2982	3014	48	244	64	479	788	5996
	192-64	3225	3138	3170	48	302	68	429	801	6308
	256-32	4785	4686	4718	48	571	133	1216	1922	9404
	256-64	4977	4893	4925	48	581	133	934	1649	9818
BIKE	128-128	5223	1541	1573	32	451	138	1684	2273	3114
	192-192	10105	3083	3115	32	1345	307	5045	6697	6198
	256-256	16494	5122	5154	32	3492	640	13990	18122	10276
Lean-BIKE	128-2-23	5223	1313	1313	32	410	88	1170	1668	2626
	192-23	8423	2533	2533	32	1380	202	3442	5024	5066
HQC	128	2321	2241	4433	32	69	138	312	519	6674
	192	4602	4514	8978	32	171	329	664	1164	13492
	256	7333	7237	14421	32	352	690	1342	2384	21658

parameter p can essentially be done freely.

Novel decoder. Figure 5 reports the comparison between the number of clock cycles taken by a straightforward bit flipping decoder such as the one described in [ABP24, ABPP26] and our proposed decoder, SLEEKDEC, for the case n_0=2 (which yields the most significant gains), and all dfr targets for SLEEK-E and SLEEK-LT. As it can be seen, trading off the computation of p upc counters for a polynomial multiplication (either over $\mathcal{R}$ or $\widetilde{\mathcal{R}}$) is advantageous for all security levels and dfrs.

5.4 Comparative Performance Analysis

We now compare the performances of the SLEEK-LT parameter sets of with the ones of BIKE and HQC: Table 3 reports the figures of merit for all our explored CCA parameter sets with $n_0 \in \{2, 5, 8\}$ (results for all n_0 values are reported in Appendix A). Moving towards higher code rates allows SLEEK to trade off an increase in public key size for a reduction in ciphertext size, gaining approximately a factor of two moving from $n_0 = 2$ to $n_0 = 8$. This allows SLEEK-LT to reach a ciphertext size of 901B for $\lambda = 128$, which is 1.7× smaller than the one of BIKE, 4.9× lower than HQC, and only 133B larger than lattice-based standard ML-KEM [Nat24a]. This size is also well below the Ethernet MTU (1500 B) and the usable payload of an UDP packet over IPv6 (1232 B), allowing the use of SLEEK in scenarios such as the Wireguard VPN protocol [HNS+21] without compromising on security guarantees against active attackers. This represents an interesting trade-off, since in IND-CCA use scenarios the public key is typically transmitted once and reused



(a) IND-CCA parameter sets

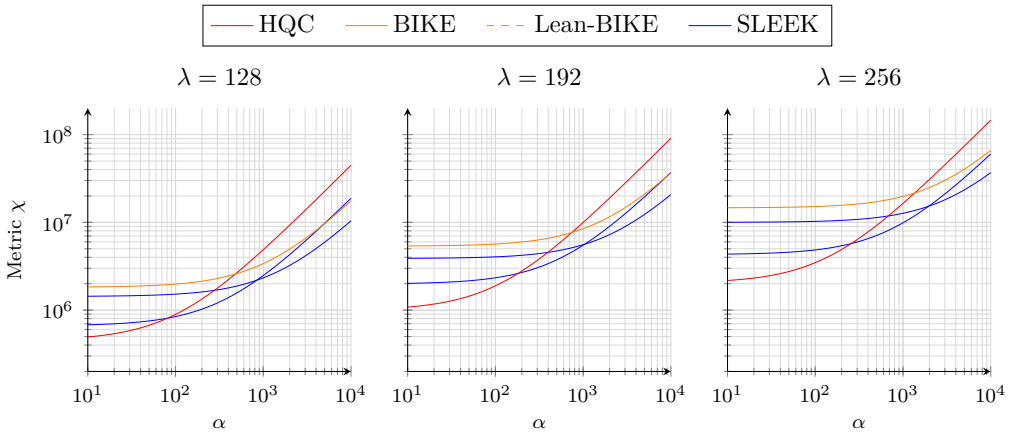
(b) IND-CPA parameter sets

Figure 6: Performance comparison among BIKE and its IND-CPA instantiation Lean-BIKE (○ marker), HQC (◊ marker), and our approach (× marker). Blue, yellow and red colors correspond to parameter sets for $\lambda = 128$, $\lambda = 192$, and $\lambda = 256$, respectively.

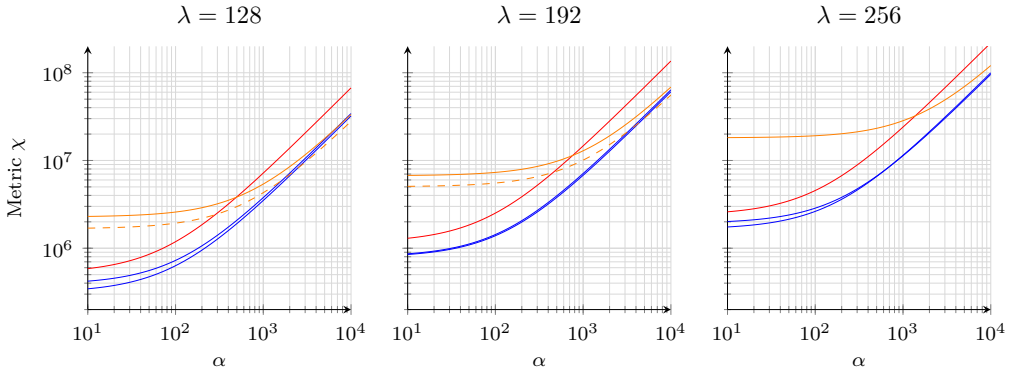
for a long time. The trade-off also increases the decryption time, making the overall KEM delay (encryption+decryption) grow in n_0: although being between 21% and 63% faster than BIKE, SLEEK-LT is between 47% and 217% slower than HQC. Finally, we note that "short-term-security", $\lambda = 80$ parameters achieve ciphertext sizes as small as 466B (roughly half of the ciphertext size for ML-KEM), and a combined encaps+decaps takes place in 245 to 468 kCC (63μs to 122μs on our platform).

Table 4 reports the figures of merit for SLEEK-E parameter sets (complete results for the constant-time implementation are in Appendix A). SLEEK-E (128-32) obtains for $\lambda = 128$ a 5.3× speedup with respect to Lean-BIKE [DKG24], the IND-CPA instantiation of BIKE, at the cost of 566B of extra bandwidth, while being $\approx 500\times$ less prone to decryption failures. Furthermore, comparing SLEEK with BIKE and HQC in the context of a drop-in use wherever IND-CPA only guarantees are required, we have that: *i)* for $\lambda = 128$, SLEEK-E (128-32) is 7.2× faster than BIKE with only 78 more bytes to be sent *ii)* for $\lambda = 128$, SLEEK-E (128-32) is 1.6× faster than HQC with a 2.1× smaller bandwidth requirement.

To provide a global outlook on the performance of SLEEK when compared to HQC, BIKE and Lean-BIKE (for its IND-CPA parametrizations) we represent the bandwidth and time requirements in two scatter plots in Fig. 6, with logarithmic x-axis for clarity (points closer to the origin are better). As it can be seen in Fig. 6 (a) reporting data for IND-CCA parametrizations, SLEEK-LT offers flexibility in trading off bandwidth for speed: markers occupying the top left position (more bandwidth, but faster) of each cluster of markers are the ones with $n_0 = 2$, while markers occupying the bottom right position are with $n_0 = 8$. Figure 6 (b) reports the figures of merit for IND-CPA parameters, showing that SLEEK-E provides significant savings in both bandwidth and computation time with respect to both HQC and BIKE. When compared to Lean-BIKE, which was specifically designed for IND-CPA scenarios, we report significant speed improvements at a small (≈ 0.5 kiB) increase in bandwidth. We also recall that Lean-BIKE targets a DFR of 2^{-23} in contrast to the 2^{-32} or 2^{-64} target of SLEEK-E. Finally, we consider a combined metric which aims at highlighting the computational efficiency of the KEM at hand when the cost of transmitting the ciphertext (and public key, for the CPA case) is taken into account. This approach was suggested on NIST's public forum during the standardization process [NIS19], with LEDAcrypt-KEM explicitly considering it in its parameter sets [BBC+20]. The idea is to adopt a combined metric $\chi = \mathrm{CC} + \alpha\mathrm{B}$ that adds together the total clock cycles count



(a) IND-CCA combined metric χ = CC Encaps + CC Decaps + $\alpha|\mathsf{ctx}|$



(b) IND-CPA combined metric χ = CC KeyGen + CC Encaps + CC Decaps + $\alpha(|\mathsf{ctx}| + |\mathsf{pk}|)$

Figure 7: Combined memory and speed metric χ evaluating the combined cost of computing the KEM and sending the ciphertext (plus the public key in the IND-CPA case), assuming that managing one byte of material to be sent takes α cycles $\chi = \text{CC} + \alpha\text{B}$.

CC for one cipher run (i.e., the cost of encapsulation and decapsulation for IND-CCA cases, and key generation, encapsulation and decapsulation for IND-CPA cases) with the amount of bytes to be sent multiplied by a scaling coefficient α. Reasonable values for α are in the 100–1000 range. Since pinpointing an exact value for α is hard, as it depends strongly on the platform at hand, we report, in Fig. 7, the χ values for $10 \leq \alpha \leq 10^4$. Figure 7 (a) reports the results for SLEEK-LT considering the fastest $n_0 = 2$ and the most compact $n_0 = 8$ parameter sets for each λ, while Fig. 7 (b) reports the results for SLEEK-E parameters. In the IND-CCA scenario SLEEK-LT is the most advantageous choice (lower values on the y axis are better) when $\alpha > 80$, $\alpha > 200$ and $\alpha > 230$ for $\lambda = 128$, 192, and 256, respectively. Furthermore, in the IND-CPA scenario SLEEK-E is the most efficient option for $\alpha < 2000$ and $\alpha < 5000$ even when compared to Lean-BIKE, which is specifically designed for IND-CPA scenarios and sacrifices the enforcement of random sampling of the PKE input to gain speed.

6 Concluding Remarks

In this work, we present SLEEK, a versatile design for a KEM with provable security guarantees and competitive performance. Our design space exploration and optimized implementation allowed us to obtain efficient KEM instances fitting both long-term key and ephemeral key usage. SLEEK-LT achieves ciphertexts up to 4.9× smaller than HQC

and execution times 1.3×–2.7× faster than BIKE, while SLEEK-E outperforms in speed even dedicated designs such as Lean-BIKE by up to 5.3×, while maintaining analytically bounded and practically negligible DFR.

Disclosure of Interests. The authors have no competing interests to declare.

References

[AAC+25] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1638–1647. ACM, 2025.

[ABB+23] Nicolas Aragon, Paulo L. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE: Bit Flipping Key Encapsulation. Submission to the NIST Post-Quantum Cryptography Standardization Process. https://bikesuite.org, 2023.

[ABC+20] Martin Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Submission to the NIST Post-Quantum Cryptography Standardization Process. https://classic.mceliece.org, 2020.

[ABE+25] Daniil Averkov, Tatiana Belova, Gregory Emdin, Mikhail Goncharov, Viktoriia Krivogornitsyna, Alexander S. Kulikov, Fedor Kurmazov, Daniil Levtsov, Georgie Levtsov, Vsevolod Vaskin, and Aleksey Vorobiev. Cirbo: A new tool for boolean circuit analysis and synthesis. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 11105–11112. AAAI Press, 2025.

[ABP24] Alessandro Annechini, Alessandro Barenghi, and Gerardo Pelosi. Bit-Flipping Decoder Failure Rate Estimation for (v, w)-Regular Codes. In *IEEE International Symposium on Information Theory, ISIT 2024, Athens, Greece, July 7-12, 2024*, pages 3374–3379. IEEE, 2024.

[ABP26] Alessandro Annechini, Alessandro Barenghi, and Gerardo Pelosi. On the Hardness of Decoding Quasi-Cyclic Codes and the Security of Code-Based Public-Key Cryptosystems. In Roberto Di Pietro, Karen Renaud, and Paolo Mori, editors, *Proceedings of the 12th International Conference on Information Systems Security and Privacy, ICISSP 2026 - Volume 2, Marbella, Spain, March 04-06, 2026*, pages 49–60. SCITEPRESS, 2026. ISBN 978-989-758-800-6, ISSN 2184-4356.

[ABPP26] Alessandro Annechini, Alessandro Barenghi, Gerardo Pelosi, and Simone Perriello. Efficient QC-MDPC cryptosystems with bounded decoding failure

rate. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part IV*, volume 16803 of *Lecture Notes in Computer Science*, pages 347–379. Springer, 2026.

[ACD+25] Gorjan Alagic, David Cooper, Quynh Dang, Nicky Jia, Jiseon Lichtinger, Dustin Moody, Ray Perlner, Daniel Smith-Tone, and Douglas Stebila. Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process. Technical Report NIST IR 8545, National Institute of Standards and Technology, March 2025.

[Ale03] Michael Alekhnovich. More on Average Case vs Approximation Complexity. In *44th Symposium on Foundations of Computer Science, FOCS 2003, Cambridge, MA, USA, October 11-14, 2003, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[ALM+25] Sarah Arpin, Jun Bo Lau, Antoine Mesnard, Ray A. Perlner, Angela Robinson, Jean-Pierre Tillich, and Valentin Vasseur. Error Floor Prediction with Markov Models for QC-MDPC Codes. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I*, Lecture Notes in Computer Science, pages 221–252. Springer, 2025.

[AMAD+24] Carlos Aguilar-Melchor, Nicolas Aragon, Jean-Christophe Deneuville, Philippe Gaborit, Amaury Métayer, and Gilles Zémor. Efficient error-correcting codes for the HQC post-quantum cryptosystem. *Designs, Codes and Cryptography*, 92(12):4511–4530, 2024.

[BBC+18] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAkem: A Post-quantum Key Encapsulation Mechanism Based on QC-LDPC Codes. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, Lecture Notes in Computer Science, pages 3–24. Springer, 2018.

[BBC+20] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAcrypt: Low-dEnsity parity-check coDe-bAsed cryptographic systems. Submission to the NIST Post-Quantum Cryptography Standardization Process, https://www.ledacrypt.org, 2020.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. 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*, pages 520–536. Springer, 2012.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, Berlin, Heidelberg, September 2007.

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory*, 24(3):384–386, 1978.

[BP05] Joan Boyar and René Peralta. The exact multiplicative complexity of the hamming weight function. *Electron. Colloquium Comput. Complex.*, TR05, 2005.

[BP18] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM Unification. Cryptology ePrint Archive, Paper 2018/526, https://eprint.iacr.org/2018/526, 2018.

[BP20] Alessandro Barenghi and Gerardo Pelosi. A comprehensive analysis of constant-time polynomial inversion for post-quantum cryptosystems. In Maurizio Palesi, Gianluca Palermo, Catherine Graves, and Eishi Arima, editors, *Proceedings of the 17th ACM International Conference on Computing Frontiers, CF 2020, Catania, Sicily, Italy, May 11-13, 2020*, pages 269–276. ACM, 2020.

[BSI26] BSI. Notes on recent developments concerning Classic McEliece. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Notes_Classic_McEliece.pdf?___blob=publicationFile&v=2, 2026.

[CCK21] Ming-Shing Chen, Tung Chou, and Markus Krausz. Optimizing BIKE for the intel haswell and ARM cortex-m4. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):97–124, 2021.

[CHLT26] Kévin Carrier, Valérian Hatey, Laura Luzzi, and Jean-Pierre Tillich. Multilevel amortized gaussian elimination in information-set decoding: Applications to HQC and PCG. Cryptology ePrint Archive, Paper 2026/1498, 2026.

[Cho16] Tung Chou. Qcbits: Constant-time small-key code-based cryptography. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, Lecture Notes in Computer Science, pages 280–300. Springer, 2016.

[CÖ09] Murat Cenk and Ferruh Özbudak. Improved polynomial multiplication formulas over if_2 using chinese remainder theorem. *IEEE Trans. Computers*, 58(4):572–576, 2009.

[Cry26] Cryptography Research Centre - Technology Innovation Institute (TII). Git Hub Repository: Crypto-TII/CryptographicEstimators - Commit 92144. https://github.com/Crypto-TII/CryptographicEstimators, 2026.

[CXK+26] Madelyn Cain, Qian Xu, Robbie King, Lewis R. B. Picard, Harry Levine, Manuel Endres, John Preskill, Hsin-Yuan Huang, and Dolev Bluvstein. Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits, 2026.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

[DG19] Nir Drucker and Shay Gueron. A toolbox for software optimization of QC-MDPC code-based cryptosystems. *J. Cryptogr. Eng.*, 9(4):341–357, 2019.

[DGK20a] Nir Drucker, Shay Gueron, and Dusan Kostic. On constant-time QC-MDPC decoders with negligible failure rate. In Marco Baldi, Edoardo Persichetti, and Paolo Santini, editors, *Code-Based Cryptography - 8th International*

Workshop, CBCrypto 2020, Zagreb, Croatia, May 9-10, 2020, Revised Selected Papers, Lecture Notes in Computer Science, pages 50–79. Springer, 2020.

[DGK20b] Nir Drucker, Shay Gueron, and Dusan Kostic. QC-MDPC decoders with several shades of gray. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, Lecture Notes in Computer Science, pages 35–50. Springer, 2020.

[DGK21] Nir Drucker, Shay Gueron, and Dusan Kostic. Fast polynomial inversion for post quantum QC-MDPC cryptography. *Inf. Comput.*, 281:104799, 2021.

[DKG24] Nir Drucker, Dusan Kostic, and Shay Gueron. A lean BIKE KEM design for ephemeral key agreement. 5th NIST PQC Standardization Conference, April 10-12, 2024, in Rockville, Maryland - https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/a-lean-bike-kem.pdf, 2024.

[Dum89] Il'ya Isaakovich Dumer. Two decoding algorithms for linear codes. *Problemy Peredachi Informatsii*, 25(1):24–32, 1989.

[Eur24] European Commission. Commission Recommendation (EU) 2024/1101 on a Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography. https://digital-strategy.ec.europa.eu/en/library/coordinated-implementation-roadmap-transition-post-quantum-cryptography, 2024.

[Eur25] European Commission and NIS Cooperation Group. A Coordinated Implementation Roadmap for the Transition to Post-Quantum Cryptography. https://digital-strategy.ec.europa.eu/en/library/coordinated-implementation-roadmap-transition-post-quantum-cryptography, 2025.

[FH07] Haining Fan and M. Anwar Hasan. Comments on "five, six, and seven-term karatsuba-like formulae'. *IEEE Trans. Computers*, 56(5):716–717, 2007.

[FKPY22] Pierre-Alain Fouque, Paul Kirchner, Thomas Pornin, and Yang Yu. BAT: Small and Fast KEM over NTRU Lattices. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):240–265, 2022.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, Lecture Notes in Computer Science, pages 53–68. Springer, 1999.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *J. Cryptol.*, 26(1):80–101, 2013.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[GdFAB19] Antonio Guimarães, Diego de Freitas Aranha, and Edson Borin. Optimized implementation of QC-MDPC code-based cryptography. *Concurr. Comput. Pract. Exp.*, 31(18), 2019.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 789–815, 2016.

[GK10a] Shay Gueron and Michael E. Kounavis. Efficient implementation of the galois counter mode using a carry-less multiplier and a fast reduction algorithm. *Inf. Process. Lett.*, 110(14-15):549–553, 2010.

[GK10b] Shay Gueron and Michael E Kounavis. Intel® carry-less multiplication instruction and its usage for computing the gcm mode. *White Paper*, 10, 2010.

[GMA+25] Philippe Gaborit, Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jérôme Lacan, Jean-Marc, Pascal Véron, Paulo L. Barreto, Santosh Ghosh, Shay Gueron, Tim Güneysu, Rafael Misoczki, Jan Richter-Brokmann, Nicolas Sendrier, Jean-Pierre Tillich, and Valentin Vasseur. Hamming Quasi-Cyclic (HQC). Submission to the NIST Post-Quantum Cryptography Standardization Process, https://pqc-hqc.org/, 2025.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.

[HHK21] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation - Errata corrige. Cryptology ePrint Archive, Paper 2017/604 - version 20211102:142303 - https://eprint.iacr.org/2017/604, 2021.

[HHMS25] Kathrin Hövelmanns, Andreas Hülsing, Christian Majenz, and Fabrizio Sisinni. (Un)breakable Curses - Re-encryption in the Fujisaki-Okamoto Transform. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II*, Lecture Notes in Computer Science, pages 245–274. Springer, 2025.

[HNS+21] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 304–321. IEEE, 2021.

[ISO10] ISO Central Secretary. Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements. Standard IEC 61508-1:2010, International Organization for Standardization, Geneva, CH, 2010.

[ISO19] ISO Central Secretary. Information security – Lightweight cryptography – Part 2: Block ciphers. Standard ISO/IEC 29192-2:2019, International Organization for Standardization, Geneva, CH, 2019.

[JZM19] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Key Encapsulation Mechanism with Explicit Rejection in the Quantum Random Oracle Model. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*, Lecture Notes in Computer Science, pages 618–645. Springer, 2019.

[Kar63] Anatolii Karatsuba. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596, 1963.

[KMSW26] Juliane Krämer, Yannick Münz, Patrick Struck, and Maximiliane Weishäupl. Binding Security of Explicitly-Rejecting KEMs via Plaintext Confirmation and Robust PKEs. *IACR Commun. Cryptol.*, 2(4):34, 2026.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116, 1978.

[Mon05] Peter L. Montgomery. Five, six, and seven-term karatsuba-like formulae. *IEEE Trans. Computers*, 54(3):362–369, 2005.

[Nat15] National Institute of Standards and Technology (NIST). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication - FIPS PUB 202 - https://doi.org/10.6028/NIST.FIPS.202-upd1, August 2015.

[Nat24a] National Institute of Standards and Technology. FIPS 203 - Module-Lattice-Based Key-Encapsulation Mechanism Standard. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf, 2024.

[Nat24b] National Institute of Standards and Technology (NIST). NIST IR 8547: Transition to Post-Quantum Cryptography Standards. https://csrc.nist.gov/pubs/ir/8547/ipd, 2024.

[Nat25] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/pqc-standardization, 2025.

[Nie86] Harald G. Niederreiter. Knapsack-Type Cryptosystems and Algebraic Coding Theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[NIS19] NIST Post quantum standardization effort mailing list. Footguns as an axis for security analysis. https://groups.google.com/a/list.nist.gov/g/pqc-forum, 2019.

[PBL25] Simone Perriello, Alessandro Barenghi, and Emanuele Lunardi. Git Hub Repository: LEDAcrypt/LEDAtools - Git Commit 4650fbf. https://github.com/tigerjack-crypto/LEDAtools, 2025.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[RP21] Mostafizar Rahman and Goutam Paul. Grover on Present: Quantum Resource Estimation. Cryptology ePrint Archive, Paper 2021/1655, https://eprint.iacr.org/2021/1655, 2021.

[Sen11] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2011.

[SF12] Chen Su and Haining Fan. Impact of Intel's new instruction sets on software implementation of GF(2)[x] multiplication. *Inf. Process. Lett.*, 112(12):497–502, 2012.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

[TFA+11a] Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López. Software implementation of binary elliptic curves: Impact of the carry-less multiplier on scalar multiplication. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, Lecture Notes in Computer Science, pages 108–123. Springer, 2011.

[TFA+11b] Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López. Speeding scalar multiplication over binary elliptic curves using the new carry-less multiplication instruction. *J. Cryptogr. Eng.*, 1(3):187–199, 2011.

[Til18] Jean-Pierre Tillich. The Decoding Failure Probability of MDPC Codes. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 941–945. IEEE, 2018.

[TMC+25] Meltem Sönmez Turan, Kerry A. McKay, Donghoon Chang, Jinkeon Kang, and John Kelsey. Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. NIST Special Publication SP 800-232 - https://doi.org/10.6028/NIST.SP.800-232, 2025.

[Uni22] United States Congress. Quantum Computing Cybersecurity Preparedness Act. Public Law No: 117-260, https://congress.gov, 2022.

[Wei26] Stephen A. Weis. Improving GIJS key recovery for classic McEliece. Cryptology ePrint Archive, Paper 2026/1984, 2026.

[WWW23] Tianrui Wang, Anyu Wang, and Xiaoyun Wang. Exploring Decryption Failures of BIKE: New Class of Weak Keys and Key Recovery Attacks. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 70–100. Springer, 2023.

[XIU+21] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-Injection Attacks Against NIST's Post-Quantum Cryptography Round 3 KEM Candidates. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, Lecture Notes in Computer Science, pages 33–61. Springer, 2021.

A Full Parameters List

Tables 5 present the parameters of all variants of SLEEK-LT. Each variant is defined by the values: λ, $n0$ (for SLEEK-LT), and $(-\log_2(\text{DFR}))$. The value p indicates the size of the circulant blocks composing the private matrix H, while v is the Hamming weight of the columns of H. The parameter t indicates the Hamming weight of the error vector $\mathbf{e}$ constituting the plaintext of the PKE. The values th_1, th_2, th_3 are the flipping thresholds employed by SLEEKDEC during the three decoding iterations.

Table 6 reports, for each SLEEK-LT parameter set, *1)* the size in bytes of keys, ciphertext, and plaintext, *2)* the number of kilo-clock-cycles (kCC) needed to generate keys, encapsulate, and decapsulate, and *3)* the sum of kCC for encapsulation and decapsulation. BIKE and HQC figures are also presented for comparison.

Table 7 is analogous to Table 6 but for the constant-time implementation of SLEEK-E. Here the last two columns report the bandwidth and time requirements when SLEEK is used as an ephemeral KEM, i.e. *1)* the sum of kCC needed for key generation, encapsulation, and decapsulation, and *2)* the sum of public key and ciphertext sizes, in bytes. Lean-BIKE is also included in the comparison.

Table 5: Code parameters and computational hardness of SLEEK-LT.

λ-$\mathbf{n_0}$-dfr	p	v	t	th_1	th_2	th_3	**Actual** dfr	λ_{cm}	λ_{ck}	λ_{qm}	λ_{qk}
80-2-80	7019	47	87	29	24	24	80.32	95.23	96.50	128.04	128.49
80-3-80	5171	53	55	32	28	27	80.84	95.23	96.34	126.89	128.16
80-4-80	4547	57	44	34	32	29	80.60	95.92	98.28	126.76	130.81
80-5-80	4091	57	38	34	32	29	83.46	96.03	95.74	126.24	128.90
80-6-80	3851	59	34	35	34	30	81.68	95.66	97.21	125.41	131.14
80-7-80	3691	59	32	35	34	30	81.07	97.47	96.15	126.97	130.69
80-8-80	3533	59	30	35	34	30	81.66	97.56	95.39	126.78	130.46
128-2-128	14341	71	136	42	37	36	128.46	143.54	143.34	179.63	178.64
128-3-128	10691	81	86	48	41	41	129.51	143.66	144.32	179.10	179.41
128-4-128	9173	85	68	50	44	43	134.80	143.22	143.74	177.72	179.33
128-5-128	8363	89	59	52	48	45	128.07	144.09	146.07	178.01	182.41
128-6-128	7691	89	53	52	48	45	128.02	144.00	143.50	177.41	180.44
128-7-128	7411	91	49	53	49	47	133.88	144.48	144.83	177.61	182.46
128-8-128	7013	91	46	53	50	46	128.02	144.82	143.56	177.64	181.69
192-2-160	27179	105	201	61	54	54	161.61	207.95	210.22	246.95	248.51
192-3-160	19997	119	127	69	61	61	160.75	208.02	209.94	246.74	247.90
192-4-160	17123	125	101	72	65	64	161.26	208.55	209.09	246.35	247.52
192-5-160	15581	129	87	74	68	66	166.89	208.43	209.46	245.64	248.58
192-6-160	14341	131	78	75	70	66	160.07	207.97	208.76	244.65	248.49
192-7-160	13613	133	72	76	71	68	160.39	208.38	209.22	244.73	249.56
192-8-160	12899	133	67	76	71	68	164.83	207.19	207.30	243.20	248.14
192-2-192	27827	105	201	61	54	53	192.42	207.98	210.24	247.07	248.61
192-3-192	20611	119	127	69	60	60	193.47	208.06	209.96	246.88	248.02
192-4-192	17597	125	101	72	64	64	192.97	208.58	209.12	246.48	247.61
192-5-192	15923	129	87	74	67	66	192.13	208.46	209.47	245.74	248.65
192-6-192	14741	131	78	75	68	67	192.54	208.01	208.78	244.79	248.58
192-7-192	13997	133	72	76	69	69	193.71	208.42	209.24	244.86	249.65
192-8-192	13229	133	67	76	69	69	194.54	207.22	207.32	243.32	248.22
256-2-160	42101	137	265	79	71	69	160.62	271.55	273.48	312.53	313.80
256-3-160	30931	155	168	89	80	79	160.64	272.54	272.38	313.57	312.35
256-4-160	26309	163	133	93	85	83	160.20	272.11	271.46	312.16	311.83
256-5-160	23899	169	115	97	87	85	160.98	272.99	273.14	312.45	314.18
256-6-160	22003	171	103	98	88	87	162.77	272.10	271.21	311.07	312.84
256-7-160	20899	175	95	100	91	89	160.74	272.45	273.90	311.09	316.16
256-8-160	19813	175	89	100	91	89	160.88	272.69	271.34	310.98	314.09
256-2-256	44531	137	265	78	70	70	257.07	271.60	273.54	312.80	314.02
256-3-256	32717	155	167	88	79	79	256.97	271.04	272.44	312.25	312.55
256-4-256	28051	163	133	92	84	83	256.35	272.19	271.51	312.46	312.06
256-5-256	25579	169	115	95	87	86	258.81	273.07	273.19	312.77	314.41
256-6-256	23549	171	103	96	89	87	256.61	272.19	271.26	311.40	313.06
256-7-256	22469	175	95	99	89	89	260.33	272.58	273.95	311.44	316.39
256-8-256	21269	175	89	99	89	89	256.12	272.78	271.39	311.32	314.31

Table 6: Comparison of performance between SLEEK-LT, BIKE and HQC.

Scheme	λ-$\mathbf{n_0}$-dfr	sk	pk	ctx	ptx	Kg	Enc	Dec	Enc+ Dec
			(Bytes)				(kCC)		
SLEEK	80-2-80	937	878	902	32	167	40	204	245
	80-3-80	705	1294	671	32	148	35	263	298
	80-4-80	633	1707	593	32	161	39	343	382
	80-5-80	569	2048	536	32	105	31	333	364
	80-6-80	545	2410	506	32	107	31	365	396
	80-7-80	521	2772	486	32	116	32	409	441
	80-8-80	505	3094	466	32	120	33	435	468
SLEEK	128-2-128	1857	1793	1817	32	451	88	573	662
	128-3-128	1401	2674	1361	32	400	80	812	892
	128-4-128	1209	3441	1171	32	430	89	989	1079
	128-5-128	1105	4184	1070	32	445	100	1217	1318
	128-6-128	1025	4810	986	32	256	73	1242	1316
	128-7-128	985	5562	951	32	284	75	1223	1299
	128-8-128	937	6139	901	32	276	77	1353	1430
SLEEK	192-2-160	3481	3398	3430	48	1285	211	1757	1968
	192-3-160	2585	5000	2532	48	1090	193	2516	2710
	192-4-160	2225	6423	2173	48	1141	227	3343	3571
	192-5-160	2033	7792	1980	48	654	168	2408	2576
	192-6-160	1881	8965	1825	48	631	178	2833	3011
	192-7-160	1785	10212	1734	48	675	187	3303	3490
	192-8-160	1697	11291	1645	48	690	198	3648	3846
	192-2-192	3561	3479	3511	48	1299	213	1770	1984
	192-3-192	2665	5154	2609	48	1067	205	2918	3124
	192-4-192	2281	6600	2232	48	1169	231	3386	3617
	192-5-192	2073	7964	2023	48	651	168	2430	2598
	192-6-192	1929	9215	1875	48	679	178	2854	3033
	192-7-192	1833	10500	1782	48	680	187	3284	3471
	192-8-192	1737	11578	1686	48	709	200	3678	3878
SLEEK	256-2-160	5345	5263	5295	48	3081	408	3796	4204
	256-3-160	3953	7734	3899	48	1540	302	3743	4046
	256-4-160	3377	9867	3321	48	1466	314	4878	5192
	256-5-160	3073	11952	3020	48	1408	324	5982	6307
	256-6-160	2833	13755	2783	48	1463	363	7126	7489
	256-7-160	2697	15678	2645	48	1513	410	8769	9180
	256-8-160	2561	17339	2509	48	1583	437	9065	9502
	256-2-256	5649	5567	5599	48	3198	419	3874	4293
	256-3-256	4177	8180	4122	48	1643	313	4094	4408
	256-4-256	3593	10521	3539	48	1522	339	5717	6056
	256-5-256	3281	12792	3230	48	1590	357	6244	6602
	256-6-256	3025	14720	2976	48	1485	371	7268	7639
	256-7-256	2897	16854	2841	48	1532	406	8555	8961
	256-8-256	2745	18613	2691	48	1599	442	9557	10000
BIKE	128-2-128	5223	1541	1573	32	451	138	1684	1822
	192-2-192	10105	3083	3115	32	1345	307	5045	5352
	256-2-256	16494	5122	5154	32	3492	640	13990	14630
HQC	128	2321	2241	4433	32	69	138	312	450
	192	4602	4514	8978	32	171	329	664	993
	256	7333	7237	14421	32	352	690	1342	2032

Table 7: Comparison of performance between a constant-time implementation of SLEEK-E, BIKE, Lean-Bike, and HQC.

Scheme	λ-dfr	sk	pk (B)	ctx	ptx	Kg	Enc (kCC)	Dec	KeyGen+ +Enc+Dec	pk+ ctx
SLEEK	80-32	849	777	801	32	57	29	190	277	1578
	80-64	921	852	876	32	61	28	191	281	1728
	128-32	1641	1584	1608	32	120	59	506	685	3192
	128-64	1761	1683	1707	32	161	65	586	813	3390
	192-32	3081	2982	3014	48	254	126	1352	1733	5996
	192-64	3225	3138	3170	48	316	137	1523	1978	6308
	256-32	4785	4686	4718	48	595	267	2709	3572	9404
	256-64	4977	4893	4925	48	604	265	2743	3613	9818
BIKE	128-128	5223	1541	1573	32	451	138	1684	2273	3114
	192-192	10105	3083	3115	32	1345	307	5045	6697	6198
	256-256	16494	5122	5154	32	3492	640	13990	18122	10276
Lean-BIKE	128-23	5223	1313	1313	32	410	88	1170	1668	2626
	192-23	8423	2533	2533	32	1380	202	3442	5024	5066
HQC	128	2321	2241	4433	32	69	138	312	519	6674
	192	4602	4514	8978	32	171	329	664	1164	13492
	256	7333	7237	14421	32	352	690	1342	2384	21658