Abstract—The evolution of quantum computers poses a serious threat to contemporary public-key encryption (PKE) schemes. To address this impending issue, the National Institute of Standards and Technology (NIST) is currently undertaking the Post-Quantum Cryptography (PQC) standardization project intending to evaluate and subsequently standardize the suitable PQC scheme(s). One such attractive approach, called Bit Flipping Key Encapsulation (BIKE), has entered the final round of the competition. Despite having some attractive features, the IND-CCA security of BIKE depends on the average decoder failure rate (DFR), a higher value of which can facilitate a particular type of side-channel attack. Although BIKE adopts the Black-Grey-Flip (BGF) decoder that offers a negligible DFR, the effect of weak-keys on the average DFR has not been fully investigated. In this paper, we implement the BIKE scheme, and then through extensive experiments show that the weak-keys can be a potential threat to IND-CCA security of the BIKE scheme and thus need attention from the relevant research community. We also propose a key-check algorithm that can potentially supplement the BIKE mechanism and prevent users from adopting weak-keys.

Index Terms—BIKE, black-gray flip decoder, code-based cryptosystems, post-quantum cryptography, weak-key analysis.

I. INTRODUCTION AND BACKGROUND

K

EY exchange mechanisms (KEMs) play a critical role in the security of the Internet and other communication systems. They allow two remote entities to securely agree on a symmetric key without explicit sharing, which can be subsequently used to establish an encrypted session. The currently used KEMs are primarily based on the RSA [1] or ECC [2] schemes whose underlying security relies on the difficulty of either integer factorization or discrete logarithm problems. However, these problems can be solved in polynomial time by quantum computation models [3] with the so-called Cryptographically Relevant Quantum Computers (CRQC) [4]. Thus, it is believed that the current PKE schemes (and KEMs, consequently) will be insecure in the post-quantum era [5], [6].

To secure KEMs, tremendous research efforts have been made to design quantum-safe PKE schemes. Currently, NIST is also undertaking a standardization project for quantum-safe KEMs. Of numerous approaches, code-based cryptosystems (CBC) are considered a promising alternative to the existing PKE schemes [7], [8]. Based on the theory of error-correcting codes, their underlying security relies on the fact that decoding a codeword without the knowledge of the encoding scheme is an $NP$-complete problem [9]. The idea of CBC was incepted by McEliece in 1978 [10], which has remained secure against classical and quantum attacks but at the cost of a large key size. To reduce the key size, Misoczki et al. proposed Quasi-Cyclic Moderate Density Parity-Check (QC-MDPC) codes and utilized them to develop the QC-MDPC variant of the McEliece scheme [11]. This variant has received much attention because of its comparable security with significantly smaller key sizes. The Bit Flipping Key Encapsulation (BIKE) [12] mechanism that is submitted to NIST for standardization as a quantum-safe KEM, is built on top of the QC-MDPC variant. Due to its promising security and performance features, BIKE has been selected in the final round of the NIST standardization competition as an alternate candidate [13]. In addition, BIKE has been recently added to the list of supported KEM schemes for the post-quantum TLS protocol used in the AWS Key Management Service (KMS) offered by Amazon [14].

Unlike the original McEliece scheme, the QC-MDPC variant and BIKE leverage a probabilistic and iterative decoder in their decryption module. The original design of the QC-MDPC variant employed the orthodox Bit-Filling (BF) decoder [15] with slight modifications [11]. However, the BF decoder (as a probabilistic and iterative decoder) suffers from higher DFR. Specifically, the decoder suffers from poor decoding performance when the number of iterations is restricted for performance considerations like accelerating the encoder. It fails in the decryption/decapsulation process, which degrades the performance and may also make the system vulnerable to side-channel/reaction attacks. For example, Guo et al. [16] introduced an efficient reaction attack for the QC-MDPC variant known as the GJS attack. In a GJS attack, the attacker first
sends crafted ciphertexts to the victim Alice while observing the reaction of her decoder for every ciphertext (i.e., successful or failure). Then, utilizing the correlation between the faulty ciphertext patterns and Alice’s private key, the attacker can fully recover her private key. Further, Nilsson et al. [17] proposed a novel technique for fast generation of the crafted ciphertexts to improve the efficiency of the GJS attack. These attacks can be regarded as a weaker version of Chosen-Ciphertext Attacks (CCA) since the adversary only needs to observe the decoder’s reaction without access to the full decryption oracle (i.e., it does not analyze any decrypted plaintext).

To tackle these attacks, the Fujisaki-Okamoto (FO) CCA transformation model has been adopted in BIKE [18], [19]. The FO model introduces a ciphertext protection mechanism to restrict the GJS attacker’s ability to craft ciphertext, when the receiver can check the integrity of any received ciphertext. Although the FO model significantly mitigates the threat of reaction attacks, the scheme must deploy a decoder with a negligible DFR to provide the indistinguishability under Chosen-Ciphertext Attack (IND-CCA) security. Sendrier and Vasseur [20] argued that to provide \(\lambda\)-bit of IND-CCA security, the average DFR (taken over the whole key space) must be upper bounded by \(2^{-\lambda}\). For this reason, several modifications of the BF decoding algorithm have been proposed to offer negligible DFR [20], [21], [22], [23], [24], [25]. For example, the latest version of BIKE deploys the Black-Grey-Flip (BGF) decoder [24] that is the state-of-the-art variant of the BF algorithm. BGF uses only five iterations for decoding while offering a negligible DFR.

However, there are some (private) key structures for which the probabilistic decoders show poor performance in terms of DFR [23], [26]. They are referred to as weak-keys since they are potentially at risk from disclosure through side-channel/reaction attacks (such as a GJS attack [16], [17]). Although the number of weak-keys is much smaller than the size of the entire key space, their effect on the average DFR must be analyzed to ensure they do not endanger the IND-CCA security of the scheme. In this regard, Drucker et al. [23], argued that IND-CCA security of BIKE cannot be claimed without proving (or disproving) the existence of weak-keys and formally quantifying their impact on DFR. Subsequently, Sendrier and Vasseur [26] have recently conducted some weak-key analysis for the QC-MDPC-based schemes and showed that the average DFR is not notably affected by the weak-keys (that have been identified so far). Based on their empirical results, they argue that the existence of weak-keys is not critical to IND-CCA claims.

However, most prior works (except [27]) do not consider the state-of-the-art BGF decoder in their analysis. For example, Sendrier and Vasseur [26] studied the previous version of the BIKE scheme that was submitted to the second round of the NIST competition (i.e., BIKE-1). In BIKE-1, the BackFlip decoder [28] was deployed to provide 128-bit security after 100 iterations (i.e., BackFlip-100). Moreover, the number of iterations has been limited to 20 for time-saving (significantly more than the BGF decoder) in existing experiments. With the compensation of a few iterations, Sendrier’s results are compared with 97-bit security, i.e., the estimated security of BackFlip-20. As an exception, Vasseur [27] attempted to fill this gap, but multiple factors (e.g., \(r\) and \(f\) parameters — see details of these parameters in Sections II-A and III.C.1) that impact the weak-key analysis remain occluded, thereby warranting a thorough analysis of weak-keys in the context of the state-of-the-art BGF decoder (see Section VI for more details).

In view of the aforementioned discussion, the existent analysis on weak-keys does not extend to the latest version of BIKE that adopts the contemporary BGF decoder. Therefore, it is important to investigate the impact of weak-keys on the latest version of BIKE. Motivated by this research gap, we implemented the BIKE scheme in Matlab and conducted extensive experiments on the basis of the IND-CCA security model presented in [18]. Our experiments show that the contribution of weak-keys in the average DFR of the BIKE’s BGF decoder is more significant than the maximum allowed level needed for the IND-CCA security. As a result, the negative effect of weak-keys on the average DFR cannot be ignored and must be addressed before claiming the IND-CCA security of BIKE. To address the weak-keys issue, we propose a key-check mechanism that can be integrated into the key generation module of the BIKE scheme to ensure the private keys generated by users are not weak. The main contributions of this paper are summarized as follows:

- Through extensive experiments and the formal model for proving the IND-CCA security, we show that weak-keys’ negative effect on average DFR is greater than the maximum allowed level. It may put the IND-CCA security of the BIKE mechanism at risk.
- We propose a key-check algorithm that can be integrated into the key generation subroutine of BIKE to ensure that users do not generate and adopt weak (private) keys.
- We perform an implementation of the BIKE scheme with the state-of-the-art BGF decoder and provide some technical key points required to implement the BIKE scheme.

The rest of this paper is organized as follows. In Section II, we provide the preliminaries required for understanding the working principles of the BIKE scheme. Section III presents the structure of weak-keys in the BIKE scheme and an intuitive understanding of their effect on IND-CCA security. In Section IV, we present the results of our experimental evaluation. After introducing the key-check mechanism in Section V, we further discuss the weak-key issues in Section VI. We conclude the paper in Section VII.

II. Preliminaries

In this section, we present the basic concepts that will help the readers understand the other sections of the paper. We first briefly review the QC-MDPC variant of the McEliece scheme (we refer the readers to [11] and [29] for more information about the McEliece scheme and its QC-MDPC variant). Then, we review the QC-MDPC PKE scheme and finally, the latest version of the BIKE scheme is described.
A. QC-MDPC Codes

Before we review QC-MDPC codes, we present some key concepts and definitions in the field of error correction codes. Error correction codes are widely used in communication protocols and recording systems to ensure reliable data transmission and storage. Considering a block of \( k \) information bits, the error correction code \( C(n, k) \) computes \( r = n - k \) redundancy bits (based on some encoding equations) and creates an \( n \)-bit block of data (called a codeword) consisting of \( k \) information bit and \( r \) redundancy bits. The codeword is subsequently sent to the relevant destination, which exploits the redundant bits (based on some decoding rules) for detecting and correcting any errors in the received message and successfully retrieves the actual information.

**Definition 1 (Linear Block Code [30]):** \( C(n, k) \) is a linear error correction code if the modulo-2 sum (i.e., XOR binary operation) of any two or multiple codewords is a valid codeword.

**Definition 2 (Hamming Weight [31]):** The Hamming weight of a codeword is defined as the number of non-zero bits.

**Definition 3 (Generator Matrix [30]):** The linear block code \( C(n, k) \) has a generator matrix \( G \in \mathbb{F}_2^{k \times n} \) which defines the one-to-one mapping between the \( k \)-bit message block \( m \in \mathbb{F}_2^k \) and the corresponding \( n \)-bit codeword \( c \in \mathbb{F}_2^n \), i.e., \( c_1x^n = m_1x^kG_{k \times n} \).

Thus, \( G \) is used by the encoder to generate the message block \( m \). The number of valid codewords for \( C(n, k) \) is \( 2^k \) which can be much smaller than \( 2^n \) (since \( n > k \)), i.e., every binary vector over \( \mathbb{F}_2^n \) is a valid codeword of \( C \). Thus, \( C \) can be considered as a \( k \)-dimensional subset of \( \mathbb{F}_2^n \).

**Definition 4 (Systematic Code [32]):** \( C(n, k) \) is called a systematic code if its generator matrix \( G \) is written in the form of \( G = [I_k|A_{k \times r}] \) in which \( I_k \) is a \( k \times k \) identity matrix and \( A \) is a \( k \times r \) coefficient matrix.

**Definition 5 (Parity-Check Matrix [30]):** The parity-check matrix \( H \in \mathbb{F}_2^{r \times n} \) of a linear code \( C(n, k) \) is \( r \times n \) matrix that is orthogonal to all the codewords of \( C(n, k) \). The syndrome vector of the block is \( r = n - k \) parity-check (redundant) bits.

**Definition 6 (Syndrome Decoding (SD) Problem [33]):** Given the parity-check matrix \( H \in \mathbb{F}_2^{r \times n} \) and the syndrome vector \( S \in \mathbb{F}_2^r \), the SD problem searches for a vector \( e \in \mathbb{F}_2^n \) with the Hamming weight \( \leq t \) such that \( S = eH^T \).

The SD problem was proved to be \( \mathcal{NP} \)-complete if the parity-check matrix \( H \) is random [33]. It establishes the essential security feature required by code-based cryptosystems to be quantum-resistant. Quantum computation models are considered to be unable to efficiently solve \( \mathcal{NP} \)-complete problems [34].

**Definition 7 (Quasi-Cyclic (QC) Code [35]):** The binary linear code \( C(n, k) \) is QC if there exists an integer \( n_0 < n \) such that every cyclic shift of a codeword \( c \in \mathbb{C} \) by \( n_0 \) bits results in another valid codeword of \( C \).

In a systematic QC code, each codeword \( e \) consists of \( n_0 \) blocks of \( n \) bits, i.e., \( n = n_0p \). Hence, every block includes \( k_0 = k/p \) information bits and \( r_0 = n_0 - k_0 \) parity bits. In a QC code \( C \) with \( r_0 = 1 \), we have

\[
r = (n - k) = (n_0 - k_0)p = r_0p = p
\]

In this case, it is shown that the parity-check matrix \( H \) of \( C \) is composed of \( n_0 \) circulant blocks of size \( p \times p \) (or \( r \times r \), equivalently) [35] which is written as

\[
H = [H_0 \ H_1 \ldots H_{n_0-1}]
\]

where each circulant block \( H_i \) has the following format:

\[
H_i = \begin{bmatrix}
    h_0^{(i)} & h_1^{(i)} & h_2^{(i)} & \ldots & h_{r-1}^{(i)} \\
    h_1^{(i)} & h_0^{(i)} & h_1^{(i)} & \ldots & h_{r-2}^{(i)} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    h_{r-1}^{(i)} & h_{r-2}^{(i)} & h_{r-3}^{(i)} & \ldots & h_0^{(i)}
\end{bmatrix}
\]

For a given \( C(n, k) \) with \( H \) as the parity-check matrix, the following generator matrix \( G \) of the above QC code \( C \) can be written as follows:

\[
G = [I_k|Q_{k \times r}] = \begin{bmatrix}
    I_k \\
    \vdots \\
    (H_{n_0-1}^{-1}H_{n_0-2})^T \\
    (H_{n_0-1}^{-1}H_{n_0-3})^T \\
    \vdots \\
    (H_{n_0-1}^{-1}H_0)^T
\end{bmatrix}
\]

The above format can be proved using the fact that \( HG^T = 0 \) and by performing some linear algebra operations on it.

**Definition 8 (QC-MDPC Codes [11]):** An \( (n, r, w) \)-QC-MDPC code is a QC code of length \( n = n_0r \) and dimension \( k = n - r = kr \) whose parity-check matrix has a constant row weight of \( w = O(\sqrt{n}) \).

We consider those QC-MDPC codes where \( r_0 = 1 \), i.e., \( r = p \). The most important characteristic of QC-MDPC...
codes is that the circulant blocks in the parity-check matrix can be described by their first row only ($r$ bits only). Thus, to construct $H$, one needs only the first row of the $n_0$ circulant blocks. Moreover, the parity-check matrix has a relatively small Hamming weight (i.e., $w \ll n$). Therefore, instead of storing $n = n_0r$ bits, the positions (indexes) of $w$ non-zero bits can be used to store $H$. These are the key features of QC-MDPC codes that enable them to significantly mitigate the key size issue of the original McEliece scheme. As described in the next subsection, a private key of the QC-MDPC variant (and BIKE, consequently) is the parity-check matrix of the selected code. In the next subsection, we briefly review the new variant of the McEliece scheme built on QC-MDPC codes.

B. The QC-MDPC PKE Scheme

The QC-MDPC PKE cryptosystem was proposed by Misoczki et al. [11] to mitigate the key size problem of the original McEliece scheme. It consisted of three subroutines, i.e., Key Generation, Encryption, and Decryption (see Fig. 1).

1) Key Generation:

a) Private key: In QC-MDPC PKE, the parity-check matrix $H$ of the underlying $(r, n, w)$-QC-MDPC code plays the role of a private key. It has the format shown in Eq. (1). To generate $H$, $n_0$ circulant blocks of size $r \times r$ must be generated. For each block $H_i$, $(0 \leq i \leq n_0 - 1)$, a random sequence of $r$ bits and Hamming weight $w_i$ is generated such that $\sum_{i=0}^{n_0-1} w_i = w$. This sequence becomes the first row of $H_i$. The remaining $r - 1$ rows are computed through cyclic shifts of the first row (i.e., $j$ cyclic shifts to generate row $j$, $1 \leq j \leq r - 1$).

$w \log_2(r)$ bits are needed to store the private key because each circulant block $H_i$ is represented by its first row only. The row can be stored using the indexes of its $w_i$ non-zero bits. Hence, $w \log_2(r)$ bits are much less than $n^2 + k^2 + nk$ bits needed to store the private key used in the original McEliece scheme.

b) Public key: The generator matrix $G$ of the underlying $(r, n, w)$-QC-MDPC code is the public key derived from the private key using Eq. (3). Since $G$ is quasi-cyclic (similar to the circulant blocks in $H$), it can be represented by its first row only, which has $n = k + r$ bits. The first $k$ bits belong to the identity matrix $I_k$ that do not need to be stored (always have a specific format). Thus, $r$ bits are required to store the public key, resulting in a significant reduction in the key size compared with $nk$ bits in the original McEliece scheme. Unlike $H$, the first row of $G$ does not necessarily have a small (and fixed) Hamming weight. Thus, the idea of storing the indexes of non-zero bits cannot be applied to store the public key.

2) Encryption: The encryption of a plaintext message $m \in \mathbb{F}_2^n$ is performed using the following equation:

$$x = m \cdot G \oplus e = c \oplus e,$$

where $e \in \mathbb{F}_2^r$ is a random vector of weight $t$ that is determined based on the error correcting capability of the corresponding decoder.

3) Decryption: The receiver performs the following procedure to decrypt the received ciphertext $x \in \mathbb{F}_2^n$:

- Apply $x$ to the corresponding $t$-error correcting decoder $\psi_H$ that leverages the knowledge of $H$ for efficient decoding. The decoder finds the error vector $e$ and returns the corresponding codeword $c = m \cdot G$.
- Return the first $k$ bits of $c$ as the decoded plaintext message $m$ because $G$ is in the systematic form.

The systematic form of $G$ may result in the scheme being vulnerable to chosen ciphertext and message recovery attacks. The reason is that the initial $k$ bits of ciphertext $x = m \cdot G \oplus e$ includes a copy of the plaintext $m$ with some possible flipped bits (since $x_i = m_i \oplus e_i$ for $1 \leq i \leq k$). In fact, two ciphertexts $x_1$ and $x_2$ are mostly distinguishable if an attacker knows their corresponding plaintexts $m_1$ and $m_2$. In the worst case, if $e_i = 0$ for $1 \leq i \leq k$, the ciphertexts are certainly distinguishable. To address this issue, a CCA transformation model can be used (e.g., [36]) that converts the plaintext to a random vector whose observation gives no useful knowledge for a CCA attacker.

Regarding the above mentioned decoder $\psi_H$, several decoding algorithms have been proposed [15], [20], [21], [22], [23], [24], [25]. We refer the readers to [23] and [24] for more information about efficient QC-MDPC decoders.

C. The BIKE Scheme

BIKE [12] leverages the QC-MDPC PKE scheme for encryption/decryption. BIKE has recently qualified as an alternate candidate for the final round of the NIST standardization project. In previous rounds of the NIST competition, BIKE was submitted in the form of three different versions (denoted by BIKE-1, BIKE-2, and BIKE-3), each of which satisfied the needs of a specific group of cryptographic applications (e.g., bandwidth, latency, security, etc.). However, in the final round, following the recommendation of NIST, it was submitted in the form of a single version that relies heavily on BIKE-2. The final version recommends three sets of system

---

**TABLE I**

| Parameter | Description | Value |
|-----------|-------------|-------|
| $r$       | Size of circulant blocks in the parity-check matrix $H$ | Level 1: 12,323, Level 2: 24,659, Level 3: 40,973 |
| $w$       | Row weight of the parity-check matrix $H$ | 142, 206, 274 |
| $t$       | Hamming weight of the error vector | 134, 199, 264 |
| $l$       | Size of the generated symmetric key $K_s$ | 256, 256, 256 |
parameters to satisfy the three different security levels defined by NIST, i.e., level 1 (128-bit security), level 2 (192-bit security), and level 3 (256-bit security) (see Table I). To address the IND-CCA security issues in the QC-MDPC variant, the Fujisaki-Okamoto (FO) CCA transformation model has been integrated into the BIKE scheme [18], [19] (see Fig. 2). In the final version of BIKE, the state-of-the-art BGF decoder [24] has been deployed in the decapsulation subroutine to provide negligible DFR in five iterations.

BIKE includes three subroutines: key generation, encapsulation, and decapsulation (see Fig. 2). The procedure is initiated by Bob who wants to establish an encrypted session with Alice. They need to securely share a symmetric key to start their encrypted session. To do this, Bob generates his public and private keys by running the key generation subroutine. Bob sends his public key to Alice who uses it to compute the ciphertext $C = (C_0, C_1)$ and the symmetric key $K_s$ (using the encapsulation subroutine). The first part of $C$ (i.e., $C_0$) is computed by encrypting the hash of a random vector $m$ using the underlying $(r, n, w)$-QC-MDPC scheme while the second part (i.e., $C_1$) protects $C_0$ against malicious manipulations. Then, Alice sends $C$ to Bob through an insecure channel. By running the decapsulation subroutine, Bob applies $C_0$ to the corresponding QC-MDPC decoder (i.e., the BGF decoder) to decrypt the data. Bob also checks the integrity of $C_0$ using $C_1$. Finally, Bob could derive the same symmetric key $K_s$ as Alice computed.

In BIKE, all the circulant matrix blocks (e.g., $H_0$ and $H_1$ of the parity-check matrix) are treated as polynomial rings since it increases the efficiency of the computations required in the key generation, encapsulation, and decapsulation subroutines. In this regard, considering $a$ as the first row of the $r \times r$ circulant matrix $A$, the $r$-bit sequence $a$ can be represented by the polynomial $(a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1}) \in \mathcal{R} = \mathcal{F}_2[x]/(x^r - 1)$ (see [28] for more information). In the following, we briefly review the three subroutines of the BIKE scheme. We refer readers to [12] for detailed information about BIKE.

1) Key Generation:

   a) Private key: Since BIKE works based on the QC-MDPC variant, its private key is the parity-check matrix of the underlying $(r, n, w)$-QC-MDPC code with $m_0 = 2$ circulant blocks. To generate the private key, two random polynomial $h_0$ and $h_1$ are generated with the Hamming weight of $|h_0| = |h_1| = w/2$. Then, $\sigma$, a random sequence of $l$ bits is generated. Finally, the private key is concatenated using the three components, denoted as $sk = (h_0, h_1, \sigma)$.

   b) Public key: Derive the public key as $pk = h = h_1 h_0^{-1}$.

2) Encapsulation: The encapsulation subroutine takes the public key $h$ as input and generates the ciphertext $C$ and the symmetric key $K_s$. Three hash functions (modeled as random oracles) are defined and used here — $H : \{0, 1\}^l \rightarrow \{0, 1\}^{2r}$, $L : \{0, 1\}^{2r} \rightarrow \{0, 1\}^l$, and $K : \{0, 1\}^{2r+l} \rightarrow \{0, 1\}^l$. The following procedure is performed in this subroutine.

   1) Randomly select an $l$-bit vector $m$ from the message space $M = \{0, 1\}^l$.
   2) Compute $(e_0, e_1) = H(m)$ where $e_0$ and $e_1$ are error vectors of $r$ bits such that $|e_0| + |e_1| = t$.
   3) Compute $C = (C_0, C_1) = (e_0 + e_1, m \oplus L(e_0, e_1))$ and send it to the recipient.
   4) Compute $K_s = K(m, C)$ as the secret symmetric key.

3) Decapsulation: The decapsulation subroutine takes the private key $sk$ and ciphertext $C$ as input and generates the symmetric key $K_s$ as follows.

   1) Decode $C_0$ by computing the syndrome $S = C_0 h_0$ and apply it to the corresponding BGF decoder to obtain the error vectors $e_0$ and $e_1$.
   2) Compute $m' = C_1 \oplus L(e'_0, e'_1)$. If $H(m') \neq (e'_0, e'_1)$, set $m' = \sigma$.
   3) Compute $K_s = K(m', C)$.

The deployed CCA transformation model prevents a CCA attacker (e.g., a GIS attacker) from freely choosing any error vector $(e_0, e_1)$ required by the attack procedure and submitting the obtained crafted ciphertext to the receiver (i.e., to craft a ciphertext based on a malicious plan). It is because the error vectors $(e_0, e_1)$ are computed by the one-way hash function $H$. If the attacker changes the legitimate error vector $e = (e_0, e_1)$, the integrity check at the receiver will fail (i.e., $H(m') \neq (e'_0, e'_1)$). Therefore, to feed the ciphertext with a desired error vector $e$, the attacker has to find the corresponding vector $m$ such that $H(m) = e$. This requirement imposes a heavy burden on the attacker since many queries must be submitted to the random oracle $H$ to identify the corresponding vector $m$. We will discuss this problem in the next section.

III. WEAK-KEYS IN THE BIKE SCHEME

In this section, we first present a formal definition for the weak-keys that we consider in this work. Then, we discuss
the effect of weak-keys on the IND-CCA security of BIKE and review the weak-key structures that have been identified so far.

A. Definition of Weak-Keys

In this work, we consider a private key of the BIKE scheme as weak if decoding the ciphertexts generated using its corresponding public key results in a much higher DFR than the average DFR of the decoder. Regardless of performance degradation issues, such weak-keys can offer a significant advantage to CCA adversaries for conducting a reaction attack (see the next subsection for more details). Prior research has indicated the possibilities of recovering weak private keys from the corresponding public key in the QC-MDPC PKE scheme [37], [38]. Specifically, it is shown that there exist weak private keys whose structure can facilitate an adversary in compromising them by applying linear algebra techniques like the extended Euclidean algorithm on their corresponding public keys. However, those weak structures are not relevant to IND-CCA security (the adversary does not need to conduct a chosen ciphertext attack to compromise those keys) and thus are not considered in this work. Instead, we assume that recovering the private key from the corresponding public key is infeasible in the BIKE scheme. Therefore, the attacker needs to conduct a chosen ciphertext attack to recover the private key by leveraging the decoder’s reactions.

1) The Negative Effect of Weak-Keys on DFR: To gain an intuitive insight into understanding the impact of weak-keys on a decoder’s performance, we need an elaboration on the following question. How do the columns of $H$ with a large intersections result in a higher DFR? To answer this question, we consider $H$ as the parity-check matrix (i.e., private key) in which columns $j$ and $l$ have $m$ non-zero bits at exactly the same positions (i.e., $m$ intersections between the two columns). If $H$ is a normal key (i.e., not a weak-key), the largest possible value of $m$ is usually small (e.g., 5) as compared with the Hamming weight of each column, i.e., $w/2$ [26]. Now, assume we have the private key $H$ in which $m$ (for $j$th and $l$th columns) is much larger than that of normal keys. Also, assume that $e_j = 0$ and $e_l = 1$ are in the original error vector $e$. In this case, it is intuitive to imagine that the number of unsatisfied parity-check equations (i.e., $upc_j$) for $j$th and $l$th bits will be highly correlated since they would have similar connections on the corresponding Tanner graph. In other words, both $j$ and $l$ bit nodes are involved in almost the same parity-check equations due to the large intersection between them. Thus, in a decoding iteration, if a specific parity-check equation (that involves bit node $j$ and $l$) is unsatisfied, it will be counted towards both $j$th and $l$th bits. Thus, it is highly likely that the decoder returns $e_j = e_l = 1$ due to their (correlated) $upc$ being greater than the set threshold. In fact, a real error at $l$th bit results in a situation that convinces the decoder to incorrectly consider $e_j$ as a set error bit. Again, in the next iteration, the same procedure is performed, this time the $j$th bit that was mistakenly considered as a set error bit (i.e., the decoder flipped it to $e_j = 1$ in the previous iteration while its real value is 0) results in a high value for the correlated $upc_j$ and $upc_l$. Thus, the decoder (again) identifies both of them as set error bits and flips their value. Although this corrects $e_j$, it results in an incorrect value for $e_l$. This process is repeated back and forth for all iterations, making the decoder incapable of finding the correct vector $e$ causing decoding failure.

B. Weak-Keys and IND-CCA Security of BIKE

As stated earlier, BIKE deploys a probabilistic and iterative decoder with a fixed number of iterations to ensure that constant-time implementation is needed for suppressing any side-channel knowledge available to an adversary [23]. This decoder can fail to successfully decode a ciphertext in the allowed number of iterations. The decoding capability of such probabilistic decoders is generally represented in terms of (average) DFR. Prior research has shown that the higher DFR of a decoder deployed in QC-MDPC-based schemes (such as BIKE) can facilitate the efficient recovery of private key through an attack referred to as GJS (attack) [16]. In this attack, some specific formats for the error vector $e$ are used to craft a large group of ciphertexts that are submitted to the decryption oracle (this constitutes CCA). Then, utilizing decryption failures, the attacker can recover the private key. As mentioned before, to circumvent this possibility, the BIKE mechanism adopts the FO CCA transformation model [18] to prevent the attacker from cherry-picking the desired error vector $e$ needed for a successful GJS attack. In BIKE, $e$ is the output of a one-way hash function to restrict attackers from crafting some specifically chosen ciphertexts. However, despite this simple remedy, a lower DFR is still important for ensuring the required level of security (see below for details).

For a formal proof of IND-CCA security, we need to define a $\delta$-correct KEM scheme. According to [18], a KEM scheme is $\delta$-correct if

$$Pr[\text{Decaps}(C, sk) \neq K_j(sk, pk) \leftarrow \text{Key_Gen}, (C, K_e) \leftarrow \text{Encaps}(pk)] \leq \delta$$

(4)

The term on the left-hand side of the aforementioned inequality is the average DFR (hereinafter denoted as $DFR$) taken over the entire key space and all the error vectors. Thus, if $DFR \leq \delta$, the KEM scheme is regarded as $\delta$-correct.

For a $\delta$-correct KEM scheme, the advantage of an IND-CCA adversary $A$ is upper bounded by

$$Adv^{CCA}_{KEM}(A) \leq q \cdot \delta + \beta,$$

(5)

where $q$ is the number of queries that $A$ needs to submit to the random oracle model (i.e., the hash function $H$) to find the valid vector $m$ that are needed for the desired error vector $e = (e_0, e_1)$ (see Fig. 2). Because $\beta$ in Eq. (5) is not relevant to IND-CCA analysis, we do not consider $\beta$ in this paper.

Based on the above definitions, a KEM scheme is IND-CCA secure offering $\lambda$-bit security if $\frac{T(A)}{Adv^{CCA}_{KEM}(A)} \geq 2^\lambda$ [18],

where $T(A)$ is the running time of $A$ that is approximated as $q \cdot t_q$, with $t_q$ representing the running time of a single query. Typically, $t_q < 1$, we have $T(A) < q$, such that $2^\lambda \leq \frac{T(A)}{Adv^{CCA}_{KEM}(A)} \leq q \cdot \delta$, resulting in $2^\lambda \leq q \cdot \delta$. Therefore,
to provide $\lambda$-bit of IND-CCA security, the KEM scheme must be $\delta$-correct with $\delta \leq 2^{-\lambda}$, or equivalently (using Eq. (4)),

$$\overline{DFR} \leq 2^{-\lambda}. \quad (6)$$

However, for the BGF decoder used in the latest version of BIKE, the accurate calculation of $\overline{DFR}$ (taken over the whole key space and error vectors) is a very challenging task. To address this issue, Baldi et al. [39] have proposed a numerically-aided approach for DFR analysis in QC-MDPC codes. However, the proposed model is based on the Maximum Likelihood (ML) and the original Bit Flipping (BF) decoders, which are not used in the latest version of BIKE. In fact, the proposed model provides a preliminary quantitative analysis of the performance gap between BF decoders and their ideal ML version based on different code parameters [39]. Thus, further research needs to be carried out to develop a precise model that can accurately estimate the DFR in BGF decoders.

In the previous research works, $\overline{DFR}$ corresponding to the needed security level is estimated through experiments with limited instances of ciphertexts (though sufficiently large) before applying (linear) extrapolation (see [20] for more details). As a result, the claimed DFR may not necessarily be the same as the actual $\overline{DFR}$. In the worst case, we assume that there is a group of keys for which the value of DFR is high (i.e., the set of weak-keys $K_w$). If such weak-keys are not used in the experiments performed for estimating the average DFR, then the actual $\overline{DFR}$ may be greater than the estimated DFR (obtained empirically) such that the condition in Eq. (6) is not met. Therefore, the impact of weak-keys $K_w$ on $\overline{DFR}$ must be investigated to estimate the actual value of average DFR and ensure the scheme’s IND-CCA security. In another NIST code-based PQC candidate called LEDAcrypt [40] based on Quasi-Cyclic Low Density Parity-Check (QC-LDPC) codes, a mathematical model has been introduced to compute worst case estimates for the DFR of the two decoders proposed in LEDAcrypt (i.e., in-place and out-place Bit Flipping decoders). It enables the efficient selection of code parameters to achieve the required DFR (associated with the desired level of security). Thus, unlike BIKE, DFR upper bound estimation is not an issue in LEDAcrypt.

To formulate the equation for IND-CCA security in the presence of weak-keys, we consider $|K_w|$ as the size of $K_w$ (i.e., the number of weak-keys), $DFR_w$ as the average DFR taken over $K_w$, $K_s$ as the set of other keys (i.e., $K_w \cup K_s$ is equal to the whole key space $K$), and $DFR_s$ as the average DFR taken over $K_s$. In this case, $\overline{DFR}$ becomes:

$$\overline{DFR} = \eta_s DFR_s + \eta_w DFR_w, \quad (7)$$

where $\eta_s = \frac{|K_s|}{|K|}$ and $\eta_w = \frac{|K_w|}{|K|}$. By combining Eqs. (6) and (7), we have,

$$\eta_s DFR_s \leq 2^{-\lambda} - \eta_w DFR_w. \quad (8)$$

From (8), the modified condition for IND-CCA security is obtained as follows:

$$\eta_w DFR_w \leq 2^{-\lambda}. \quad (9)$$

Therefore, to provide $\lambda$-bit of IND-CCA security, the set of weak-keys $K_w$ must be negligible enough (compared with $|K|$) such that $\eta_w DFR_w < 2^{-\lambda}$ (even if $DFR_w$ is significantly greater than $DFR_s$).

### C. Structure of Weak-Keys in BIKE

The weak-keys for the BIKE scheme can be determined by adopting the approach proposed in [16] (i.e., the GJS attack methodology for recovering private keys). In this attack, the attacker primarily targets $H_0$ (i.e., the first block of the private key). If $H_0$ is successfully recovered, then the attacker can easily compute $H_i$ from $H_0$ by performing simple linear algebra operations on $G.H^T = 0$. Precisely, using Eq. (3), we re-write $G.H^T = 0$ as $[k_i]Q_{x NOTES} [H_0 H_1]^T = 0$, resulting in $H_0^T + Q.H_1^T = 0$, since $k_i.H_0^T = H_0^T$. Therefore, the attacker can easily obtain $H_1 = [Q^{-1}.H_0^T]^T$ (for $n_0 = 2$, we have $r = k$).

To find $H_0$, the attacker selects the error vector $e$ from a special subset $\Psi_d$ ($d = 1, 2, \ldots, U$). The parameter $d$ is defined as the distance between two indexes (positions) $i$ and $j$ in the first row of $H_0$ (denoted by $h_0$), which is formally defined as follows:

$$d(i, j) = \min((i - j + r) \mod r, (j - i + r) \mod r) \quad \text{for } i, j \in \{0, 1, 2, \ldots, r - 1\}.$$ 

For example, considering $r = 10$, the distance between the first and last bits is 1 because $d(0, 9) = \min((19 \mod 10), (1 \mod 10)) = \min(9, 1) = 1$. Based on the above definition, $\Psi_d$ is generated as follows:

$$\Psi_d = \{e = (e_0, e_1) \mid e_0 = 0, \exists \{p_i\}_{i=1}^{t} \text{ s.t. } e_{p_i} = 1, \text{ and } p_{2i} = (p_{2i-1} + d) \mod r \text{ for } i \in \{1, 2, \ldots, t/2\} \}.$$ 

The second half of the error vector $e$ in $\Psi_d$ is an all-0 vector, i.e., the Hamming weight of $e_0$ is $t$. Each $p_i$ ($i \in \{1, 2, \ldots, t\}$) indicates the position of a non-zero bit in $e_0$.

According to [16], when the error vectors are selected from $\Psi_d$, a strong correlation exists between the decoding failure probability and the existence of distance $d$ between the non-zero bits in $h_0$ (i.e., the first row of first circulate block $H_0$). In other words, if distance $d$ exists between two non-zero bits in $h_0$, the probability of decoding failure is much smaller in contrast with the case when such a distance $d$ does not exist. Utilizing this important observation, the attacker (empirically) computes the probability of decoding failure for various distances $d = 1, 2, \ldots, U$. This attack is made by submitting many ciphertexts (generated using the error vectors selected from $\Psi_d$) to the decryption oracle and recording the corresponding result of the decryption (i.e., successful or failed). Each value of $d$ is either [existing] or [not existing] depending on the obtained failure probability. If a $d$ value has a small failure probability, it is categorized as existing; otherwise, if the $d$ value has a large failure probability, it is categorized as not existing. Based on the categorized distances $\{d\}$, the distance spectrum of $h_0$ is defined as follows:

$$D(h_0) = \{d : 1 \leq d \leq U, d \in \{\text{existing}\} \text{ in } h_0\}.$$
Moreover, since a distance \( d \) may appear multiple times in \( h_0 \), the multiplicity of \( d \) in \( h_0 \) is defined as follows:
\[
\mu(d, h_0) = |\{(i, j) : 0 \leq i \leq j \leq r - 1, \ h_{0,i} = h_{0,j} = 1 \text{ and } d(i, j) = d\}|.
\]
Finally, based on the obtained distance spectrum \( D(h_0) \) and the multiplicity of every distance \( d \in D(h_0) \), the attacker can reconstruct \( h_0 \). To do this, the attacker assigns the first two non-zero bits of \( h_0 \) at position 0 and \( d_1 \), where \( d_1 \) is the minimum distance in \( D(h_0) \). Then, the third non-zero bit is put (iteratively) at a position such that the two distances between the third position and the previous two positions exist in \( D(h_0) \). This iterative procedure continues until all the \( w/2 \) non-zero bits of \( h_0 \) are placed at their positions. The attacker needs to perform one or multiple cyclic shifts on the obtained vector to find the actual vector \( h_0 \). The first non-zero bit was placed at position 0, but it is not necessarily the case in \( h_0 \).

The structure of weak-keys in BIKE are determined based on the concepts introduced in the GJS attack (i.e., distance, distance spectrum, and multiplicity of a distance). In this regard, three types of weak-keys specified in [26] are detailed below.

1) Type 1: Using the polynomial representation of binary vectors, the Type 1 weak-key (denoted by \( h = (h_0, h_1) \)) with \( f \)-consecutive non-zero positions and Hamming weight \( w \) is defined in [26] as follows:
\[
h_i = \phi_d(x^l[(1 + x + x^2 + \ldots + x^{f-1}) + h'_i]), \quad i \in \{0, 1\},
\]
where \( d \in \{1, 2, 3, \ldots, \lfloor r/2 \rfloor \} \) is the distance between non-zero bits of the \( f \)-bit pattern in the weak-key, \( l \in \{0, 1, 2, \ldots, r - 1 \} \) determines the beginning position of the \( f \)-bit pattern, and \( \phi_d(t) \) is a mapping function that replaces \( x \) with \( x^d \), resulting in distance \( d \) between any two successive 1’s in \( (1 + x + x^2 + \ldots + x^{f-1}) \).

To construct the Type 1 weak-keys in this format, each block \( h_i; \ i \in \{0, 1\} \) (with the Hamming weight of \( w/2 \)), is divided into two sections. The first section is an \( f \)-bit block in which all the \( f \) bits are set to 1 (i.e., \( 1 + x + x^2 + \ldots + x^{f-1} \), in the polynomial form). The second section is \( h'_i \) which is an \((r - f)\)-bit block with the Hamming weight \((w/2 - f)\) and randomly chosen nonzero bits. Then, the two sections are concatenated into one piece before applying an \( f \)-bit cyclic shift using the \( x^f \) term. Finally, by applying \( \phi_d \), the \( f \)-consecutive non-zero bit of the first section are mapped to a block of \( f \)-consecutive non-zero bits. In this type of weak-key, considering the distances \( j \) (\( j \in \{1, 2, 3, \ldots, f - 1\} \)), a lower bound for the multiplicity metric \( \mu \) can be obtained as \( \mu(jd, h_0) \geq f - j \) for \( d \in \{1, 2, 3, \ldots, \lfloor r/2 \rfloor \} \).

To compute \( |K_{w/2}| \), firstly, consider the second section of a weak-key defined using Eq. (10), i.e., \( h'_i \). It is a \( r - f \)-bit vector of the Hamming weight \( w/2 - f \). Thus, we have \( \binom{w/2 - f}{r - f} \) options for \( h'_i \). For the first part of the weak-key (i.e., the \( f \)-bit pattern), there is only one option as all the \( f \) bits are set to 1. The entire (concatenated) package is subsequently shifted (cyclically) by \( l \) bit, where \( l \in \{0, 1, 2, \ldots, r - 1\} \).

Thus, we have \( r \) different options for the cyclic shifts. Finally, there can be \( \lfloor r/2 \rfloor \) different mappings for \( \phi_d(t) \), as \( d \in \{1, 2, 3, \ldots, \lfloor r/2 \rfloor \} \). Consequentially, for \( |K_{w/2}| \), we have [26]:
\[
|K_{w/2}| = 2(\binom{r - f}{w/2 - f} - \binom{r - f}{w/2 - f - 1}).
\]

The second term is essential in Eq. (11) because there are two circulant blocks in \( h \). Finally, \( \eta_w \) for this type is obtained as follows:
\[
\eta_w(f) = |K_{w/2}| = 2(\binom{r - f}{w/2 - f} - \binom{r - f}{w/2 - f - 1}).
\]

2) Type 2: According to [26], Type 2 weak-keys have a single distance with a high multiplicity factor. The Type 2 weak-key (denoted by \( h = (h_0, h_1) \)) with the Hamming weight \( w \) and parameter \( m \) has the multiplicity \( \mu(d, h_0) \) for a distance \( d \in \{1, 2, 3, \ldots, \lfloor r/2 \rfloor \} \) and \( i \in \{0, 1\} \). If \( m = w/2 - 1 \), the number of Type 2 weak-keys \( |K_{w/2}| \) is upper bounded by \( 2(\binom{r - f}{w/2 - f}) \) because the distance between all the \( w/2 \)-non-zero bits of \( h \) is \( d = w/2 - 1 \). Unlike the first type, the \( h \) blocks \( (i \in \{0, 1\}) \) do not have a second section \( h'_i \). Thus, the term \( \binom{r - f}{w/2 - f} \) in Eq. (11) is replaced with 1.

However, for \( m < w/2 - 1 \), the upper bound for \( |K_{w/2}| \) is obtained using a more complicated approach. Consider the general format \( (z_1, o_1, z_2, o_2, \ldots, z_r, o_r) \) for \( h \ (i \in \{0, 1\}) \) that starts with \( z_1 \) 0's followed by \( o_1 \) 1's followed by \( z_2 \) 0's, etc., \( (z_i, o_i > 0,i > 0,i\sum_{t=1}^{s} o_t = w/2 \) and \( s = r - w/2 \). According to [26], the upper bound for \( |K_{w/2}| \) is as follows:
\[
|K_{w/2}| \leq 2(\binom{r - f}{w/2 - f} - \binom{r - f}{w/2 - f - 1}) = \binom{r - f}{w/2 - o_1 - 1} \binom{w/2 - o_1 - 1}{s - 2}.
\]

where \( s \) is the number of \( z_i \) and \( o_i \) blocks. Eq. (13) can be proven by applying the stars and bars principle [41] on the sets of \( \{o_i\}_{i=1}^{s} \) and \( \{z_i\}_{i=1}^{s} \) separately. Following this principle, the number of \( b/t \)-uples of positive integers \( (x_1, x_2, \ldots, x_b) \) whose sum is \( N \) (i.e., \( \sum_{t=1}^{b} x_i = N \)) is \( \binom{N+b-1}{b-1} \).

Considering the set \( \{o_i\}_{i=1}^{s} \), the value of \( o_1 \) (the number of bits in the first container) varies from \( 1 \) to \( m + 1 \) and for each value of \( o_1 \) the principle is applied on the remaining \( s - 1 \) containers, i.e., \( b = s - 1 \) (we need to consider the condition \( o_1 \leq m + 1 \) to satisfy \( \mu(1, h_0) = m \)). Therefore, for every value of \( o_1 \), we have \( b = s - 1 \) and \( N = w/2 - o_1 \), as \( \sum_{t=1}^{s} o_t = w/2 \) and \( o_1 \) 1 bits are already allocated to the first container. It results in the term \( \binom{w/2 - o_1}{s - 2} \) in Eq. (13). Similarly, for the set \( \{z_i\}_{i=1}^{s} \), for every value of \( z_1 \) (\( 1 \leq z_1 \leq r - w/2 + m + 1 \)), we have \( b = s - 1 \) and \( N = r - w/2 - z_1 \) that results in the term \( \binom{r - w/2 - z_1 - 1}{s - 2} \) in Eq. (13). The term \( (o_1 + z_1) \) is applied to consider the number of different circular shifts applicable in each case.

3) Type 3: Unlike the previous types that consider a single block of the parity-check matrix, Type 3 weak-keys are defined such that the columns of \( h_0 \) and \( h_1 \) (in the parity-check matrix) jointly create ambiguity for the BF-based decoder, resulting in a high DFR. If column \( j \) of \( h_0 \) and column \( l \) of \( h_1 \) have \( m \) non-zero bits at exactly the same positions
(i.e., $m$ intersections between the two columns) and $m$ is large, their number of unsatisfied parity-check (i.e., $upc_j$ and $upc_i$) equations (counted during the decoding procedure) will be highly correlated. In this case, if $e_j = 1$ or $e_i = 1$ in the real error vector $e$, the high level of correlation can prevent the decoder from finding $e$ within the allowed number of iterations.

According to [26], the upper bound for $|K_w|$ in Type 3 weak-keys is obtained as follows:

$$|K_w| \leq r \binom{w/2}{m} \left( \frac{r - m}{w/2 - m} \right).$$

(14)

Firstly, $m$ positions should be chosen from the set of $w/2$ positions of the non-zero bits. Then, once the positions of $w/2 - m$ position can be chosen from the remaining $r - m$ available positions. Finally, in every case, $r$ circular shifts of the obtained vector are also weak-keys, resulting in the term $r$ in Eq. (14). There is no term 2 in Eq. (14) (compared with Eqs. (11) and (13)) because the second block of $h$ follows the same structure as the first block such that $[h_0 \ast h_1] = m$ for $l \in \{0, 1, 2, \ldots, r-1\}$, where $\ast$ indicates component-wise product.

IV. EXPERIMENTAL SETUP

In this section, we provide technical details about our BIKE implementation. Then, we present the results of our extensive experiments and provide a relevant discussion.

A. Our Implementation

We implemented the key generation, encryption, and decoding modules of the BIKE scheme in Matlab.1 We used the BIKE system parameters $(r, w, t) = (13232, 142, 134)$ suggested in the latest technical specification of BIKE [28] for $\lambda = 128$ bit security level. The simulations were performed on eight powerful servers equipped with Intel(R) Xeon(R) 2.5GHz CPU (6 processors) and 64GB RAM. We used the extrapolation technique proposed in [20] to estimate the DFR of the BGF decoder for the weak-keys. In the following, we provide technical details on implementing key generation, encryption, and decoding procedures.

1) Key Generation: The most difficult challenge in the implementation of the key generation module is to perform the polynomial inversion operation (over $\mathbb{F}_2$) which is needed to compute the public key from a private key (see Eq. (3) and Section II-C.1). Due to the large value of $r$, computing the inverse in the matrix domain is an inefficient approach, and thus can contribute towards computational overhead for our analysis. To have the light-weight inverse operation for our analysis, we adopted the latest extension of the Itoh-Tsujii Inversion (ITI) algorithm [42], [43] for polynomial inversion. The ITI algorithm is based on Fermat’s Little Theorem that gives the inverse of polynomial $a = (a_0 + a_1x + a_2x^2 + \ldots + a_{r-1}x^{r-1})$ as $a^{-1} = a^{2^{r-1}-2}$. The ITI algorithm provides an efficient calculation of $a^{-1}$ through $i$ cyclic shifts of the $a$’s binary vector. To utilize this approach, the adopted extension of the ITI algorithm uses a novel technique to convert $2^{r-1} - 2$ to a series of $2^i$ sub-components that are computed using easy-to-implement cyclic shifts.

Based on the mentioned approach, the adopted algorithm needs to perform $\left| \log(r - 1) \right| + wt(r - 2) - 1$ multiplications and $\left| \log(r - 2) \right| + wt(r - 2) - 1$ squaring operations to compute the inverse, where $wt(r - 2)$ indicates the Hamming weight of the value of $r - 2$ written in the binary format. Thus, it is a scalable algorithm in terms of $r$ and much more efficient than the matrix inverse computation.

We performed polynomial multiplications before generating the public key. For each set of experiments, we saved the generated public and private keys in a file such that the keys can be easily accessed by the encryption and decoding scripts, respectively.

2) Encryption: To compute the ciphertexts, polynomial multiplication is the basic operation performed in the encryption module (see Section II-C.2). We developed the following approach to compute the multiplication of two polynomials $a = (a_0 + a_1x + a_2x^2 + \ldots + a_{r-1}x^{r-1})$ and $b = (b_0 + b_1x + b_2x^2 + \ldots + b_{r-1}x^{r-1})$ of degree $r - 1$.

Assuming that $a \cdot b = c = \left(c_0 + c_1x + c_2x^2 + \ldots + c_{r-1}x^{r-1}\right)$,

we computed the binary coefficients of $c$ as $c_0 = a_0b_0 \oplus a_1b_{r-1} \oplus \ldots \oplus a_{r-1}b_1$, $c_1 = a_0b_1 \oplus a_1b_0 \oplus \ldots \oplus a_{r-1}b_2$, ..., $c_{r-1} = a_0b_{r-1} \oplus a_1b_{r-2} \oplus \ldots \oplus a_{r-1}b_0$.

Observe that

$$c_i = \sum_{j=0}^{r-1} a_jb_{(i+j) \mod r} \mod 2,$$  

(15)

for $i \in \{0, 1, 2, \ldots, r-1\}$.

We implemented Eq. (15) to perform the multiplications required for computing the ciphertext $C_0$. The same approach is used to perform the polynomial multiplications in the key generation module.

3) Decoding: We implemented a BGF decoder based on Algorithm 1 provided in [28] with 128 bit security level, i.e., $\lambda = 128$, $NbIter = 5$, $\tau = 3$. In each iteration, the syndrome vector is initially updated using the received ciphertext/updated error vector and the private key. Then, the number of unsatisfied parity-check ($upc$) equations for each node bit $i \in \{0, 1, \ldots, n - 1\}$ is counted and compared with the threshold $T = max(0.0069722, |S| + 13.530, 36)$, where $|S|$ is the Hamming weight of the syndrome vector updated at each iteration. If it is greater than $T$, the relevant bit $i$ in the error vector is flipped. The next iteration is executed using the updated error vector. Finally, after $NbIter = 5$ iterations, if the updated syndrome $S$ is equal to $eH^T$, vector $e$ is returned as the recovered error vector; otherwise, the decoder returns failure.

In the first iteration of the BGF decoder, two additional steps are performed that are related to the Black and Gray lists of the bit nodes. These two lists are created and maintained in the first iteration to keep track of those bit flips that are considered uncertain. The Black list includes those bit nodes that have been just flipped (i.e., $upc > T$), while the Gray list maintains the index of those bit nodes for which $upc$ is very close to the threshold such that $upc > T - \tau$. Then, to gain

---

1All Matlab scripts used in this paper are available at https://github.com/leipandeakin/BIKEWeakKeys
more confidence in the flipped bits, a Black/Gray bit node is flipped if the updated upc is greater than the empirically set threshold \((w/2 + 1)/2 + 1\).

**B. Experimental Methodology**

As mentioned before, the accurate calculation of \(\text{DFR}\) for the BGF decoder is a very challenging task. To circumvent this, in prior works, DFR is estimated empirically. We adopt a similar empirical approach for DFR estimation in this work, but with an emphasis on weak-keys, a feature that has been overlooked in prior works, specifically in the context of BGF decoders (i.e., recommended for BIKE mechanism submitted to NIST). To conduct our analysis and visualize the impact of weak-keys upon BIKE (i.e., BGF decoder), we leveraged our Matlab implementation as detailed above in Section IV-A. We started our analysis by crafting Type 1 weak-keys (see Section III-C.1) for varying values of \(r\) (this is selected such that 2 is primitive modulo \(r\)) and by incrementing parameter \(f\) from 5 to 40 in steps of 5 for each value of \(r\).

Ideally, one needs to perform an analysis on the value of \(r\) that will result in the DFR corresponding to the required level of security. This is because the average DFR must be upper bound by \(2^{-\lambda}\) for ensuring the \(\lambda\)-bit of IND-CCA security [20]. Therefore, for 128-bit of security (i.e., the minimum requirement for NIST standardization), the DFR of the deployed decoder should be less (or equal) than \(2^{-128}\). In other words, \(2^{28}\) ciphertexts must be generated and applied to the decoder to record a single failure, which is impractical even on powerful and efficient computing platforms. In view of this bottleneck, prior research (see [20]) has resorted to extrapolation techniques applied to the DFR curve obtained with some small values of \(r\) (as compared with that needed for DFR of \(2^{-128}\), but sufficiently large to estimate the overall trend of DFR). This technique is based on the assumption (supported by empirical data) that \(\log(\text{DFR}_\lambda(r))\) is a concave and decreasing function for \(\text{DFR}_\lambda(r) \geq 2^{-\lambda}\) (see the last paragraph of Section VI for further discussion on this assumption). More precisely, in this approach, DFR is empirically obtained for some smaller values of \(r\), resulting in relatively large DFRs measured using simulation. Then, the last two points on the DFR curve are linearly extrapolated to obtain the point corresponding to the desired value of \(r\) needed for the target security level (e.g., \(r = 12323\) in the BIKE BGF decoder for 128-bit security).

We adopt a similar methodology for our analysis — i.e., for each value of \(f\) (from 5 to 40), we compute DFR with two relatively small values of \(r\) before extrapolating them to \(r = 12323\) (i.e., the same \(r\) value proposed in BIKE [12] corresponding to DFR of \(2^{-128}\)). Precisely, we compute DFR at each point with at least 1,000 failures, ensuring the confidence interval of 95% [20]. Our analysis revealed that, as per expectations, \(\text{DFR}_w\) increases with \(f\), thereby allowing us to move the tested values of \(r\) from \(r_1 = 9739\) and \(r_2 = 9817\) (for \(f = 5, 10, 15,\) and 20) to \(r_1 = 10103\) and \(r_2 = 10181\) (for \(f = 25\)), and \(r_1 = 10181\) and \(r_2 = 10273\) (for \(f = 30\)). Moreover, for \(f = 35\) and \(f = 40\), since we expected large values for \(\text{DFR}_w\), we did not need to perform the extrapolation approach, i.e., DFR was directly measured at \(r = 12323\). It is noteworthy that the \(r\) values used in our work are different from the \(r\) values used in other works (e.g., in [23]). Because DFR is quadratic, our DFR estimations are not directly comparable to those reported in other articles.

Once we obtain DFR for each value of \(f\) (5-40), Eq. (9) suggests that for providing the \(\lambda\)-bit of IND-CCA security, \(P_w = \eta_w \cdot \text{DFR}_w\) must be smaller than \(2^{-\lambda}\), where \(\eta_w = |\mathcal{K}_w|\). If \(P_w\) is less than \(2^{-128}\) (i.e., for 128-bit security), then weak-keys have no significant impact on DFR and the BIKE IND-CCA security. Otherwise (i.e., if \(P_w\) is greater than \(2^{-128}\)), the IND-CCA security of BIKE is impacted by weak-keys, which is a potential concern.

**C. Results**

Fig. 3 depicts the actual \(\text{DFR}_w\) (i.e., \(\log_2(\text{DFR}_w)\)) for two values of \(r\) that we tested for each value of \(f\) (i.e., 5 to 40) along with the results of linear extrapolation for \(r = 12323\) (i.e., proposed value in BIKE [12] for 128-bit security). As shown in Fig. 3, the DFR corresponding to \(r = 12323\) (i.e., obtained through linear extrapolation) increases with \(f\) (see \(\log_2(\text{DFR}_w) = -96.28\) for \(f = 5\) vs \(\log_2(\text{DFR}_w) = -18.99\) for \(f = 30\)). For large values of \(r\), i.e., 35 and 40, the corresponding values of \(\text{DFR}_w\) are observed to be 0.8 and 1 (\(\log_2(\text{DFR}_w)\) will be -0.32 and 0), respectively, even when no extrapolation is performed and tested directly with \(r = 12323\). For gaining insights into the IND-CCA security of the BIKE mechanism in the presence of weak-keys, we are interested in finding the values of term \(P_w = \eta_w \cdot \text{DFR}_w\). In accordance with our prior discussion presented in Sections III-C.1 - III-C.3 (see Eqs. (12) - (14)), \(\eta_w\) in this equation is dependent upon \(f\) (in Type 1 weak-keys) and multiplicity parameter (in Type 2 and 3 weak-keys). The values of \(\eta_w\) for Types 1, 2, and 3 weak-keys with varying parameters (i.e., \(f\) and multiplicity factor) are computed using Eqs. (12) - (14), as shown in Fig. 4. As expected, for large parameters \(f\) and \(m\), the number of weak-keys are negligible as compared with the size of the whole key space. It results in negligible values for \(\eta_w\) at larger values of \(f\) and \(m\).

**Table II** summarizes the impact of Type 1 weak-keys on IND-CCA security of the BIKE scheme. As listed in **Table II**, for \(f = 5\), the term \(P_w = \eta_w \cdot \text{DFR}_w\) results in a value that is not appropriate for ensuring 128-bit security (i.e., the
minimum requirement for NIST standardization). Precisely, for $f = 5$, the corresponding value of $P_w$ is $2^{-106.50} \not\leq 2^{-128}$, suggesting that the corresponding security level with this parameter is inadequate for NIST standardization. In fact, the contribution of Type 1 weak-keys with $f = 5$ in the average DFR of the decoder is greater than the maximum allowed value. For re-affirming our observations, we repeated our experiments for $f = 5$, this time with $r = 10009$ which is closer to $r = 12232$ and performed the extrapolation procedure with $r = 9817$ and $r = 10009$. The result shows that DFR (at $r = 12323$) in this case increases further to $-80.5$ compared to the previously obtained value of $-96.28$. It re-confirms that the term $F_w = \eta_w.DFR_w = 2^{-90.73}$ in this case is greater than the required value of $2^{-128}$. It is noteworthy that we only obtained 13 failures out of 300,000 ciphertexts in this case (versus 1,000 in the previous case) due to a greater value of $r$. However, our repeated experiments with decoders suggest that the failure rate generally remains consistent when tested with sufficiently large numbers of ciphertexts. Moreover, the overall contribution of Type 1 weak-keys on the average DFR is the sum of values listed in the last column of Table II (as well as the values for other $f$ that we have not considered in this work, e.g., $f = 6, 7, 8$, etc.), which may further impact the overall average DFR negatively. However, only the first row of Table II suffice to demonstrate the negative effect of weak-keys on IND-CCA security of the BIKE scheme that may need further exploration. Note that, we have not repeated our experiments for Types 2 and 3 weak-keys. We expect similar results for small values of $m$ in Types 2 and 3 (e.g., $m = 8$) since their dependent parameter (i.e., multiplicity) exhibits a correlation with $f$ of Type 1 weak-keys (see Fig. 4). We leave this analysis for the future to have definitive insights.

V. A NOVEL KEY-CHECK ALGORITHM

In light of our analysis mentioned above, we propose an algorithm that can potentially supplement the BIKE mechanism in selecting private keys that are not weak and thus can aid in ensuring IND-CCA security. The proposed algorithm is based on the structure of each type of weak-keys reviewed in Section III. The algorithm is depicted in Algorithm 1.

Assume that $h = (h_0, h_1)$ indicates a private key generated by a user, where $h_i$ ($i \in \{0, 1\}$) is a polynomial with the maximum degree of $r - 1$. We consider each polynomial $h_i$ as $h_i = \sum_{j \in \text{Supp}(h_i)} x_j^j$, where $\text{Supp}(h_i) = \{p_1, p_2, \ldots, p_w\}$ is called the support of $h_i$ that includes the positions of non-zero coefficients in $h_i$ (or non-zero bits in the corresponding binary vector). The proposed key-check algorithm takes $h_i$ ($i \in \{0, 1\}$) as the input and returns either Weak or Normal as the output. It first initializes the distance vector $D = \{D_1, D_2, \ldots, D_{\lfloor r/2 \rfloor}\}$ by assigning 0 to all its elements, i.e., $D_d = 0$ ($d \in \{1, 2, \ldots, \lfloor r/2 \rfloor\}$). At the end of the algorithm, every element $D_d$ will indicate the multiplicity of distance $d$ ($d \in \{1, 2, \ldots, \lfloor r/2 \rfloor\}$) in the private key. Thus, the largest element of $D$ will be compared with a specific threshold (e.g., 5, 10, and alike) to decide whether it should be considered as Weak or Normal.
Algorithm 1 Key-Check Algorithm

Input : Multiplicity threshold $T$, $h_i = \sum_{j \in \text{Supp}(h_i)} x^j$
for $i \in \{0, 1\}$ and
$\text{Supp}(h_i) = \{p^{(i)}_1, p^{(i)}_2, \ldots, p^{(i)}_{w/2}\}$, $r$, and $w$ 

Output: Weak or Normal 

1 for $i \leftarrow 0$ to 1 do
2 \hspace{1em} $D = \{D_1, D_2, \ldots, D_{\lfloor \log r \rfloor}\} = \emptyset$; 
3 \hspace{1em} for $j \leftarrow 1$ to $w/2$ do 
4 \hspace{2em} for $k \leftarrow j + 1$ to $w/2$ do 
5 \hspace{3em} temp = distance($p^{(i)}_j$, $p^{(i)}_k$) = 
6 \hspace{4em} min($p^{(i)}_j - p^{(i)}_k + r$ mod $r$); 
7 \hspace{3em} $D_{\text{temp}} = D_{\text{temp}} + 1$; 
8 \hspace{2em} end 
9 \hspace{1em} if max($D$) > $T$ then 
10 \hspace{2em} return Weak; 
11 \hspace{1em} end 
12 \hspace{1em} end 
13 \hspace{1em} end 
14 \hspace{1em} for $k \leftarrow 1$ to $w/2$ do 
15 \hspace{2em} if $|h_0 \times x^{p^{(0)}_j - p^{(0)}_k} x^j|_1 > T$ then 
16 \hspace{3em} return Weak; 
17 \hspace{2em} end 
18 \hspace{1em} end 
19 return Normal; 

For every block $h_i$ ($i \in \{0, 1\}$), the algorithm computes the distance between the position of every non-zero coefficient (denoted by $p^{(i)}_j$) and the positions of the remaining non-zero coefficients located at the right-hand side (the for loop in line 4 starts at $j + 1$). Then, the multiplicity counter $D_{\text{temp}}$ associated with the computed distance temp is increased by 1. It is repeated for all the $w/2$ non-zero coefficients of $h_i$. Finally, if the maximum element of $D$ exceeds the multiplicity threshold $T$, the key is identified as a weak-key.

Lines 13-20 of the algorithm check the key against the weak-key structure of Type 3. In Type 3, the component-wise product of block $h_0$ and $\sigma$-bit shifts of block $h_0$ must have a size larger than $T$, where $\sigma$ is the distance between the position of non-zero coefficients in $h_0$ (specified by $p^{(0)}_j$) and the positions of non-zero coefficients in $h_1$ (specified by $p^{(1)}_k$). In fact, applying a circular shift of $p^{(0)}_j - p^{(0)}_k$ bits on $h_1$ (by applying the $x^{p^{(0)}_j - p^{(0)}_k}$ term) will move the non-zero coefficient of $h_1$ located at position $p^{(1)}_k$ to position $p^{(1)}_0$. In this case, the two polynomials ($h_0$ and $x^{p^{(0)}_j - p^{(0)}_k} h_1$) will have a non-zero coefficient at the same position $p^{(1)}_j$, potentially resulting in many intersections between the corresponding columns in the parity-check matrix. Finally, if the multiplicity of intersections is less than the threshold, the key is identified as a ‘normal’ key.

VI. FURTHER DISCUSSION

In this section, we discuss the recent research works on DFR analysis of the BIKE system.

In a recent study [27], Vasseur has studied the effect of all three factors to compute the syndrome vector on the average DFR. Generally, a syndrome vector with a low Hamming weight will make it much more difficult for the BGF decoder to successfully decode the associated ciphertext in a limited number of iterations (see Section III-A.1 for details). Thus, the factors that yield low weight syndrome vectors must be carefully studied to analyze the DFR values that are higher than the average DFR. In syndrome calculation (i.e., $S = xH^T = (c \oplus e)H^T$), three factors can lead to a syndrome vector $S$ with a low Hamming weight. These include (1) the codeword $c$, (2) the error vector $e$, and (3) the parity-check matrix $H$, which is the private key (see Section II-A for details). The effect of the first two factors (i.e., codeword $c$ and error vector $e$) on the average DFR is visualized through a near-codewords analysis, according to [44]. Precisely, the syndrome of a valid codeword is a vector with all zero bits, and thus, the vectors that result in low-weight syndrome vectors are referred to as near-codewords. In fact, near-codewords are low weight codewords or error vectors that result in a low weight syndrome. The study of the third factor and its influence on DFR is performed using the so-called weak-key analysis as investigated in this paper.

In a recent report published by Vasseur [27], it is confirmed that if the extrapolation assumption is true (i.e., given a security level $\lambda$, the DFR curve is concave for values of block size $r$ at which DFR is greater than $2^{-\lambda}$), an upper bound on DFR is obtained using the extrapolation method while the weak-keys and near-codeword analysis provide a lower bound on DFR. However, the accuracy of both methods heavily depends on the computational resources used to run the required extensive simulations [27]. Regarding the weak-key analysis, Vasseur has shown that the product of the density of weak-keys by their average DFR is below $2^{-\lambda}$, and thus the weak-keys may not negatively affect the average DFR computed over the whole key space. This is in line with our results for $f > 5$ (see Table II). However, interestingly and more importantly, for $f = 5$ (which is the main point of our work as it led to $\eta_wDFR_w = 2^{-106.50}$, which is greater than $2^{-\lambda} = 2^{-128}$), our results do not conform to Vasseur’s findings. The potential reasons for this difference could be manyfold. Firstly, it remains unclear whether the explicit analysis was performed for $f = 5$ according to [27]. Secondly, the computed value of DFR for $f = 5$ (via simulations) seems occluded in [27], thereby making it difficult to understand the reason behind this difference. Finally, unlike our work, the $r$ values used in [27] for simulation prior to extrapolation are also unclear. In view of this discussion, we maintain that a clear understanding of this difference necessitates a subsequent analysis with precise values of $r$, $f$, and the methodology adopted for computing the DFR prior to extrapolation. This analysis will assist the wider research community in narrowing down the impact of weak-keys on the security of BIKE. We leave this analysis for future work. Nevertheless, we maintain that the impact of weak-keys discussed both in our work and in [27] are
empirical and conclude to different important findings which will undoubtedly help the relevant researchers to formulate the subsequent research to understand better the BIKE scheme and its security for the post-quantum era.

In another very recent work [45], the error floor behavior of QC-MDPC codes has been analyzed at the 20-bit security level. However, the authors of [45] have not considered the weak-keys of the BIKE scheme in their analysis. Instead, they have focused on low weight codewords and error patterns which result in values of DFR that are much higher than the average DFR. More precisely, they have only studied the influence of near-codewords on the average DFR of the BIKE system. In their analysis, the authors of [45] have used three sets of near-codewords $\mathcal{C}$, $\mathcal{N}$, and $2\mathcal{N}$ proposed by Vasseur [44]. In their work, although they confirm that Vasseur’s near-codeword sets include many vectors that yield decoding failures, they have not found that the defined three sets of vectors are responsible for increased DFR. In other words, they have found that the error vectors that yield low weight syndrome vectors are not fully captured in the near codewords sets defined by Vasseur in [44]. According to their conclusion, it remains for future work to find the error vectors that result in the high decoding failures observed in their experiments [45]. IND-CCA security of the BIKE scheme is very important as it provides a more accurate selection of system parameters that result in a sufficiently low DFR.

In a very recent work, Baldi et al. [39] proposed a novel method to obtain a lower bound on the DFR curve. Baldi et al. [39] mathematically proved that the DFR curve of the Maximum Likelihood (ML) decoder (which is the optimum decoder) decays polynomially in the code length (this automatically proves the same for any other sub-optimal decoder). Hence, the existence of a floor area in the DFR curve is mathematically proven, which contradicts the extrapolation assumption made in [20]. For $\lambda = 128$, the lower bounds calculated in [39] do not contradict the results provided in [12] and [24] and in this work (obtained through extrapolation). However, if the lower bound calculated based on the method offered in [39] (for specific values of $\lambda$ and other system parameters) is greater than $2^{-\lambda}$, the DFR results obtained through extrapolation will be incorrect.

VII. CONCLUSION

This paper investigated the impact of weak-keys on IND-CCA security of the BIKE post-quantum key encapsulation mechanism. We first implemented the BIKE scheme with the parameters suggested in the BIKE technical specifications. Then, we performed extensive experiments to estimate the DFR of the BIKE BGF decoder for weak-keys. Our analysis suggests that the weak-keys are precarious for the IND-CCA security of the BIKE scheme and thus need attention from the relevant research community. We believe this issue can be addressed by a potential key-check algorithm that we propose to supplement the BIKE mechanism. Theoretically, our key-check algorithm can prevent users from adopting weak private keys. The empirical analysis of the key-check algorithm and Types 2 and 3 weak-keys are left for future work to have an affirmative understanding of the effect of weak-keys on the BIKE mechanism. In addition, the analysis of the overhead of our proposed key-check algorithm would also be a promising future research direction that may reveal the need for further optimization of our proposed algorithm.

ACKNOWLEDGMENT

The authors thank Dr. Nir Drucker (IBM Research—Haifa) for providing useful comments on the article.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[2] N. Kobitz, “Elliptic curve cryptosystems,” Math. Comput., vol. 48, no. 177, pp. 203–209, 1987.

[3] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in Proc. 35th Annu. Symp. Found. Comput. Sci., 1994, pp. 124–134.

[4] Quantum Computing and Post-Quantum Cryptography. Accessed: Jan. 29, 2023. [Online]. Available: https://bit.ly/3Huuaq

[5] M. Mosca, “Cybersecurity in an era with quantum computers: Will we be ready?” IEEE Secur. Privacy, vol. 16, no. 5, pp. 38–41, Sep./Oct. 2018.

[6] W. Barker, W. Polk, and M. Souppaya, “Getting ready for post-quantum cryptography: Explore challenges associated with adoption and use of post-quantum cryptographic algorithms,” Publications NIST Cyber Security White Paper (DRAFT), Gaithersburg, MD, USA, CSRC. NIST. GOV 26, 2020.

[7] A. Daniel et al., “Initial recommendations of long-term secure post-quantum systems,” Technische Universität Eindhoven, Eindhoven, The Netherlands, Tech. Rep. Horizon 2020 ICT-645622, 2015.

[8] T. M. Fernandez-Carames, “From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the Internet of Things,” IEEE Internet Things J., vol. 7, no. 7, pp. 6457–6480, Jul. 2020.

[9] N. Sendrier, “Code-based cryptography: State of the art and perspectives,” IEEE Secur. Privacy, vol. 15, no. 4, pp. 44–50, Aug. 2017.

[10] R. J. McEliece, “A public-key cryptosystem based on algebraic,” Coding Theory, vol. 4244, pp. 114–116, Apr. 1997.

[11] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, “MDPC-McEliece: New McEliece variants from moderate density parity-check codes,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2013, pp. 2069–2073.

[12] Official Web Page of BIKE Suite. Accessed: Jan. 28, 2023. [Online]. Available: https://bit.ly/3d515Hn

[13] D. Moody, G. Alagic, D. C. Apon, D. A. Cooper, and Q. H. Dang, “Submission support on the second round of the NIST post-quantum cryptography standardization process,” Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, 2020, doi: 10.6028/NIST.IR.8309.

[14] Round 2 Post-Quantum TLS is Now Supported in AWS KMS. Accessed: Jan. 29, 2023. [Online]. Available: https://aws.amazon.com/blogs/security/round-2-post-quantum-tls-is-now-supported-in-aws-kms/

[15] R. G. Gallager, “Low-density parity-check codes,” IRE Trans. Inf. Theory, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[16] Q. Guo, T. Johansson, and P. Stankovski, “A key recovery attack on MDPC with CCA security using decoding errors,” in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. Berlin, Germany: Springer, 2016, pp. 789–815.

[17] A. Nilsson, T. Johansson, and P. S. Wagner, “Error amplification in code-based cryptography,” Lund Univ., Lund, Sweden, Tech. Rep. techs.v2019, 2018.

[18] D. Hofheinz, N. Kößelnig, and E. Kiltz, “A modular analysis of the Fujisaki–Okamoto transformation,” in Proc. Theory Cryptogr. Conf. New York, NY, USA: Springer, 2017, pp. 341–371.

[19] N. Drucker, S. Gueron, D. Kostic, and E. Persichetti, “On the applicability of the Fujisaki–Okamoto transformation to the BIKE KEM,” Int. J. Comput. Math., Comput. Syst. Theory, vol. 6, no. 4, pp. 1–13, 2021.

[20] N. Sendrier and V. Vasseur, “About low DFR for QC-MDPC decoding,” in Proc. Post-Quantum Cryptogr. 11th Int. Conf. Post-Quantum Cryptogr. (PQCrypto), vol. 12100. New York, NY, USA: Springer, 2020, pp. 20–34.
[21] N. Sendrier and V. Vasseur, “On the decoding failure rate of QC-MDPC bit-flipping decoders,” in Proc. SPIE/International Conf. Post-Quantum Cryptogr. New York, NY, USA: Springer, 2019, pp. 404–416.

[22] N. Drucker and S. Gueron, “A toolbox for software optimization of QC-MDPC code-based cryptosystems,” J. Cryptograph. Eng., vol. 9, no. 4, pp. 341–357, Nov. 2019.

[23] N. Drucker, S. Gueron, and D. Kostic, “On constant-time QC-MDPC decoding with negligible failure rate,” Univ. Haifa, Haifa, Israel, Tech. Rep. IACR 2019/1289, 2019, p. 1289.

[24] N. Drucker, S. Gueron, and D. Kostic, “QC-MDPC decoders with several shades of gray,” in Proc. Int. Conf. Post-Quantum Cryptogr. New York, NY, USA: Springer, 2020, pp. 35–50.

[25] A. Nilsson, I. E. Bocharova, B. D. Kudryashov, and T. Johansson, “A weighted bit flipping decoder for QC-MDPC-based cryptosystems,” in Proc. Int. Workshop Code-Based Cryptogr. New York, NY, USA: Springer, 2010, pp. 62–68.

[26] N. Sendrier and V. Vasseur, “On the existence of weak keys for QC-MDPC decoding,” Univ. de Paris, France, Tech. Rep. IACR 2020/1232, 2020.

[27] V. Vasseur, “QC-MDPC codes DFR and the IND-CCA security of BIKE,” Inria, Paris, France, Tech. Rep. hal-03534003, 2022.

[28] M. Aragon et al., “BIKE: Bit-flipping key encapsulation,” Univ. Limoges, Limoges, France, Tech. Rep. hal-01671903, 2017.

[29] M. Repka and P. Zając, “Overview of the McEliece cryptosystem and its security,” Tatra Mountains Math. Pub., vol. 60, no. 1, pp. 57–83, 2014.

[30] D. J. Costello Jr., and G. D. Forney, “Channel coding: The road to channel capacity,” Proc. IEEE, vol. 95, no. 6, pp. 1150–1177, Jun. 2007.

[31] N. Bonello, S. Chen, and L. Hanzo, “Low-density parity-check codes and their rateless relatives,” IEEE Commun. Surveys Tuts., vol. 13, no. 1, pp. 3–26, Feb. 2011.

[32] Y. Fang, G. Bi, Y. L. Guan, and F. C. M. Lau, “A survey on protograph LDPC codes and their applications,” IEEE Commun. Surveys Tuts., vol. 17, no. 4, pp. 1898–2016, 4th Quart., 2015.

[33] E. Berlekamp, R. J. McEliece, and H. C. A. Van Tilborg, “On the inherent intractability of certain coding problems (Corresp.),” IEEE Trans. Inf. Theory, vol. IT-24, no. 3, pp. 384–386, May 1978.

[34] S. Aaronson, “The limits of quantum,” Sci. Amer., vol. 298, no. 3, pp. 62–69, 2008.

[35] M. Baldi, “QC-LDPC code-based cryptosystems,” in QC-LDPC Code-Based Cryptography. New York, NY, USA: Springer, 2014, pp. 91–117.

[36] K. Kobara and H. Imai, “Semantically secure McEliece public-key cryptosystem-conversions for McEliece PKC,” in Proc. Int. Workshop Public Key Cryptogr. New York, NY, USA: Springer, 2001, pp. 19–35.

[37] M. Bardet, V. Dragoi, J.-G. Luque, and A. Otmani, “Weak keys for the quasi-cyclic MDPC public key encryption scheme,” in Proc. Int. Conf. Cryptol. Afr. New York, NY, USA: Springer, 2016, pp. 346–367.

[38] N. Aydin, B. Yildiz, and S. Uludag, “A class of weak keys for the QC-MDPC cryptosystem,” in Proc. Algebr. Combinat. Coding Theory (ACCT), Oct. 2020, pp. 1–4.

[39] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, “Performance bounds for QC-MDPC codes decoders,” in Code-Based Cryptography, A. Wachter-Zeh, H. Bartz, and G. Liva, Eds. Cham, Switzerland: Springer, 2022, pp. 95–122.

[40] Official Home Page of LedaCrypt. Accessed: Jan. 28, 2023. [Online]. Available: https://www.ledacrypt.org

[41] W. Feller, An Introduction to Probability Theory and Its Applications. Hoboken, NJ, USA: Wiley, 1950.

[42] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in GF(2^n) using normal bases,” Inf. Comput., vol. 78, no. 3, pp. 171–177, 1988.

[43] N. Drucker, S. Gueron, and D. Kostic, “Fast polynomial inversion for post quantum QC-MDPC cryptography,” in Proc. Int. Symp. Cyber Secur. Cryptogr. Mach. Learn. New York, NY, USA: Springer, 2020, pp. 110–127.

[44] V. Vasseur, “Post-quantum cryptography: A study of the decoding of QC-MDPC codes,” Ph.D. dissertation, Centre Inria de Paris, Université de Paris, Paris, France, 2021.

[45] S. Arpin, T. R. Billingsley, D. R. Hast, J. B. Lau, R. Pertner, and A. Robinson, “A study of error floor behavior in QC-MDPC codes,” in Proc. Int. Conf. Post-Quantum Cryptogr. New York, NY, USA: Springer, 2022, pp. 89–103.

Mohammad Reza Nosouhi received the master’s degree in telecommunications engineering from the Isfahan University of Technology, Isfahan, Iran, in 2007, and the Ph.D. degree from the University of Technology Sydney, Ultimo, NSW, Australia, in 2020. He has more than ten years of industry experience in the ICT field. He is currently working as a Research Fellow with the Centre for Cyber Resilience and Trust (CREST), Deakin University, Australia. His research interests are post-quantum cryptography, next generation authentication systems, applied cryptography, and blockchain systems.

Syed W. Shah received the Ph.D. degree in computer science and engineering from the University of New South Wales (UNSW), Sydney, Australia, and the M.S. degree in electrical and electronics engineering from the University of Bradford, U.K. He is currently a Research Fellow with Deakin University, Australia. His research interests include pervasive/ubiquitous computing, user authentication/identification, the Internet of Things, signal processing, post-quantum cryptosystems, data analytics, privacy, and security.

Lei Pan received the Ph.D. degree in computer forensics from Deakin University, Australia, in 2008. He is currently a Senior Lecturer with the Centre for Cyber Resilience and Trust (CREST), School of Information Technology, Deakin University. He leads the research theme ‘Defensive Technologies and Information Warfare’ at CREST. His research interests cover broad topics in cybersecurity and privacy. He has authored 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Yevhen Zolotavkin received the Ph.D. degree from the University of Tampere, Finland, in 2011. He is currently a Senior Researcher with Barkhausen Institut, Dresden, Saxony, Germany. Before he joined Barkhausen Institut, he worked as a Lecturer in cyber security with the School of Information Technology, Deakin University; and a Research Fellow with the Centre for Cyber Resilience and Trust (CREST), Deakin University, and Cyber Security Cooperative Research Centre, Australia. He worked at Monash University as a Research Fellow, prior to joining Deakin University. His current research interests include authentication and self-sovereign identity. In the past, he specialized on incentives for blockchain consensus.
Ashish Nanda is currently a Research Fellow with the Centre for Cyber Resilience and Trust (CREST), Deakin University, Australia, and the Cyber Security Cooperative Research Centre (CyberCRC). His research interests include multi-model authentication techniques, distributed networks/computing, and designing security frameworks for complex systems.

Praveen Gauravaram received the Ph.D. degree in cryptology from the Queensland University of Technology, Brisbane, Australia. He has held scientific positions in India, Europe, and Australia. He is currently a Senior Scientist with Tata Consultancy Services Ltd. (TCS), Australia and New Zealand (ANZ). He primarily leads TCS’s Co-Innovation (COIN) Research and Development Partnership with Cyber Security Cooperative Research Centre (Cyber Security CRC), Australia by actively engaging with the internal and external stakeholders. He has committed to the national cyber security development activities and has made significant contributions to the Australian Cyber Security Strategy 2020, National Quantum Strategy 2022, and other related Government initiatives such as the Defense Signals Directorate (DSD), the Department of Industry, Innovation and Science (DIIS), and industry partners. He is a member of the research council of the Oceania Cyber Security Centre (OCSC) and a member of the executive council of the IoT Alliance Australia (IoTAA).

Robin Doss (Senior Member, IEEE) is currently the Director of the Centre for Cyber Resilience and Trust (CREST), Deakin University. He also leads the ‘Development of Next Generation Authentication Technologies’ theme within the national Cyber Security Cooperative Research Centre (CSCRC). He has an extensive research publication portfolio. His research interests include the broad areas of system security, protocol design, and security analysis with a focus on smart, cyber-physical, and critical infrastructures. In 2019, he was a recipient of the ‘Cyber Security Researcher of the Year Award’ from the Australian Information Security Association (AISA). His research program has been funded by the Australian Research Council (ARC), government agencies such as the Defense Signals Directorate (DSD), the Department of Industry, Innovation and Science (DIIS), and industry partners. He is a member of the research council of the Oceania Cyber Security Centre (OCSC) and a member of the executive council of the IoT Alliance Australia (IoTAA).