IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 May 2023, accepted 31 May 2023, date of publication 5 June 2023, date of current version 9 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3282928

TOPICAL REVIEW

Bit Flipping Key Encapsulation for the Post-Quantum Era

MOHAMMAD REZA NOSOUHI, SYED WAJID ALI SHAH, LEI PAN, AND ROBIN DOSS, (Senior Member, IEEE)
Centre for Cyber Resilience and Trust, School of Information Technology, Deakin University, Waurn Ponds, VIC 3216, Australia
Cyber Security Cooperative Research Centre (CSCRC), Joondalup, WA 6027, Australia

Corresponding author: Lei Pan (l.pan@deakin.edu.au)

This work was supported in part by the Cyber Security Cooperative Research Centre (CSCRC), Australia.

ABSTRACT It is a matter of time before quantum computers will break the cryptosystems like RSA and ECC underpinning today's internet protocols. As Post-Quantum Cryptography (PQC) is a low-cost approach compared to others like quantum key distribution, the National Institute of Standards and Technology (NIST) has recently reviewed and analyzed numerous approaches to PQC. As a PQC candidate, Bit Flipping Key Encapsulation (BIKE) is expected to be standardized as a general-purpose Key Encapsulation Mechanism (KEM) by NIST. However, it lacks a comprehensive review of BIKE associated with technical analysis. This paper aims to present an in-depth review and analysis of the BIKE scheme with respect to relevant attacks. We provide a comprehensive review of the original McEliece (ME) scheme and present a detailed discussion on its practical challenges. Furthermore, we provide an in-depth study on the challenges of ME and BIKE cryptosystems in achieving the Indistinguishability under Chosen-Ciphertext Attack (IND-CCA) security. We provide an analysis of these cryptosystems and their security against several attacks before pointing out the research gaps for strengthening BIKE.

INDEX TERMS Authentication, security and privacy protection, post quantum cryptography, BIKE, IND-CCA security.

I. INTRODUCTION

Public-key cryptography plays an essential role in the security and integrity of digital services and technologies like communication protocols, cryptocurrencies, cyber security applications, and many more. Generally, the security of the currently used public-key cryptosystems and KEM schemes relies upon the complexity and difficulty of computational problems such as integer factorization problems (used in RSA cryptosystem [1]) and discrete logarithm problems (used in DSA [2], ECDSA [3], El Gamal [4], Diffie-Hellman (DH) key exchange [5], and ECDH [6] algorithms). In classical computation models, both integer factorization and discrete logarithm problems are considered computationally hard. However, recent studies suggest that these "computationally hard" challenges will become solvable in polynomial time using quantum computation models [7], [8], [9] with the so-called Cryptographically Relevant Quantum Computers (CRQC) [10]. Such CRQCs will enable the implementation of Shor's quantum algorithm that solves the integer factorization and discrete logarithm problems in polynomial time [11]. CRQCs will also reduce the security level of symmetric-key encryption schemes and cryptographic hash functions through Grover's algorithm [12]. It poses a severe threat to the security of the internet and other critical ICT systems and demands immediate and incremental research efforts to secure current and future data.

Despite the potential for CRQC to break current public-key cryptosystems (asymmetric encryption algorithms), symmetric encryption and cryptographic hash functions may remain relatively secure in the post-quantum era [13], [14]. It is because the current research suggests that quantum algorithms are unlikely to efficiently solve NP-hard problems [9] that are the basis of symmetric encryption schemes and hash functions (see Section II for details about NP-hard problems). At most, the symmetric algorithms and hash

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

TABLE 1. The effect of quantum computers on the security of the commonly used cryptographic algorithms.

Type of Cryptosystem	Cryptographic algorithm	Cryptographic approach	Application	Security Posture in the Post-Quantum Era	
				Effective Quantum Algorithm	Solution
Symmetric	AES	Block Ciphers	Encryption (Confidentiality)	Grover's Algorithm [12]	Needs larger key sizes to be quantum–safe
Hash-Based	SHA-2 SHA-3	Hash functions	Digital signatures Hash-based crypto Integrity Protection	Grover's Algorithm [12]	Need larger hash lengths to be quantum–safe
Asymmetric	DSA	Discrete logarithms	Digital signatures	Shor's Algorithm [11]	Quantum–insecure (alternative quantum–safe schemes must be developed and used)
	RSA	Integer factorization	Key establishment Digital signatures		
	ECDH	Elliptic curves	Key establishment		
	ECDSA	Elliptic curves	Digital signatures		

functions will only need to increase their key size and hash length to maintain the current level of security. For example, in the post-quantum era, a 128-bit level of security can be achieved using AES-256 with a key size of 256. Similarly, a quantum birthday attack presented in [15] uses Grover's algorithm [12] to find collisions for hash functions. 171-bit security is achievable through SHA-2 and SHA-3 hash algorithms with 512 bits output [16]. The anticipated impact of envisaged quantum computers on various cryptographic algorithms is summarized in Table 1.

The security of public-key cryptosystems remains a priority in the post-quantum era. NIST [17] is undertaking a standardization project to identify suitable public-key and KEM schemes for the post-quantum era [18]. In July 2022, NIST completed the third round of the standardization project, and four KEM candidate schemes were selected for the fourth round [19] where three of them are code-based algorithms, i.e., BIKE [20], Classic McEliece (CME) [21], and Hamming Quasi-Cyclic (HQC) [22] schemes. Having proven to be quantum-resistant, code-based cryptosystems use the theory of error correcting codes (more precisely, they are based upon structural codes with some intentionally inserted errors such that only the legitimate receiver with the right knowledge could recover the plaintext). Their security inherently relies on the fact that decoding a codeword without the knowledge of the encoding scheme is an NP-complete problem [23], [24]. The security of code-based cryptosystems does not rely on the complexity of any mathematical problem that quantum computers solve efficiently.

However, code-based cryptosystems suffer from many practical issues. For example, the original McEliece (ME) cryptosystem [25] has large public and private key sizes, e.g., 500 KB or more (depending on the required level of accuracy), which is a few hundred times more than the current public-key cryptosystems with a key size of 1 KB to 2 KB. Newer variants of the McEliece cryptosystem reduce the key sizes by complicating the code structure. Unfortunately, the applied modifications incurred security compromises, resulting in successful attacks [26], [27]. The BIKE scheme has been developed to mitigate the key size issue while preserving the security guarantees. Therefore, BIKE is expected to be selected as a standard general-purpose KEM at the end of Round 4 of NIST's PQC standardization process [19].

Although recent studies in [28] and [29] analyze code-based cryptosystems, the literature lacks a comprehensive study of detailed attack descriptions with respect to the latest version of BIKE. We aim to provide insights into BIKE's IND-CCA security and related issues. We adopt a review-cum-tutorial approach to contribute to a deep understanding of the working principles of code-based cryptosystems and present an overview of the known attacks and how they can potentially impact IND-CCA security. In addition, the provided details are important for system architects, security solution designers, and developers to adopt the BIKE cryptosystems in real-world applications. Considering the adoption of standardized technology is crucial as it may take years to integrate the standardized algorithm into various applications successfully. Moreover, our analysis can be used as a reference for comparing with other types of post-quantum public-key cryptosystems.

Our main contributions are summarized as follows:

- We present an in-depth review of the state-of-the-art BIKE scheme along with the necessary preliminaries.
- We comprehensively analyze ME and BIKE and describe the reasoning behind the success/failures of various attacks. We also describe the working principles of Bit Flipping Decoder and its state-of-the-art instantiation adopted in BIKE.
- We analyze the IND-CCA security of the ME and BIKE cryptosystems, provide an in-depth discussion on the impact of Decoder Failure Rate (DFR) on IND-CCA security, and point out future research directions.

In the remainder of the paper, Section II provides preliminaries on complex problems and code-based cryptography. Section III presents a brief overview of the orthodox ME cryptosystem. Section IV details QC-MDPC-based variant, decoding principles, BIKE, and relevant attacks. Finally, Section V concludes the paper and identifies future research directions.

II. PRELIMINARIES

This section presents the foundation for the next sections of the paper. We first briefly review the categories of computationally-hard problems. Then, we review the syndrome decoding problem, a fundamental element in the security of code-based cryptosystems. We refer the readers to [30] and [31] for detailed information about the theory of error correcting codes.

Throughout this paper, we use bold upper-case letters to denote matrices (e.g., $\boldsymbol{H}$). For vectors (matrices with a single row or column), we use bold lower-case letters (e.g., $\boldsymbol{e}$).

A. COMPLEXITY OF PROBLEMS

Computational problems are classified based on their inherent complexity. In the following paragraphs, we review various categories in the literature [32], [33].

1) P PROBLEMS

The set of computational problems solvable by algorithms that terminate in some steps bounded by a polynomial in the length of the input, e.g., shortest path problem [34], Rubik's puzzle [35], and many alike.

2) NP PROBLEMS (NON-DETERMINISTIC IN POLYNOMIAL TIME)

The set of verifiable problems in polynomial time (if we are given a correct answer). The NP category covers a range of problems that differ regarding their inherent hardness level. In the easiest form, they include the P category. Based on the definitions, any P problem is NP since it can be verified in polynomial time. On the other hand, the NP class covers algorithmic problems with a high hardness level (with no known deterministic solution that terminates in polynomial time). The computational problems used in the currently deployed cryptosystems are predominantly NP problems with various hardness levels [36], [37]. In addition to the hardness property, they need to be verifiable by legitimate parties in polynomial time. For example, there is no known classical algorithm to solve the factorization problem (used in the RSA cryptosystem) in polynomial time (i.e., the hardness feature). However, the party with the private key (a solution to the problem) can efficiently decrypt the ciphertext (i.e., the verification property).

3) NP-HARD PROBLEMS

The set of problems that have (at least) the same hardness as that of the most difficult problem(s) in the NP category. Moreover, verifying a given answer to the NP-hard problem in polynomial time is challenging. It suggests an overlap between the NP and NP-hard classes, thereby defining another class called NP-complete.

4) NP-COMPLETE PROBLEMS

The set of problems that are both NP and NP-hard. They have two fundamental features: (i) a high level of hardness to solve in polynomial time (as an NP-hard problem), and (ii) verifiable in polynomial time (as an NP problem).

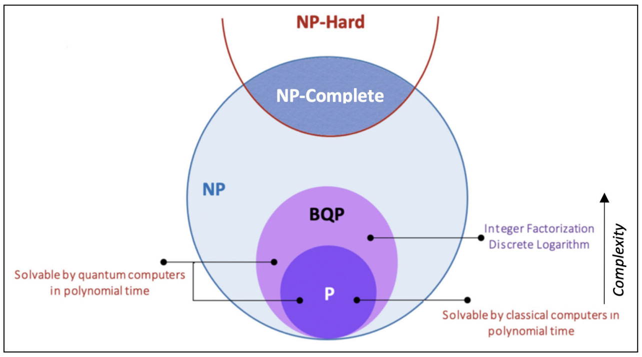

NP-Hard
NP-Complete
NP
BQP
P
Integer Factorization
Discrete Logarithm
Complexity
Solvable by quantum computers in polynomial time
Solvable by classical computers in polynomial time

FIGURE 1. NP-Complete problems are outside the BQP class, meaning that quantum computers can not solve them in polynomial time. However, popular cryptosystems' (RSA and DSA) security relies on integer factorization and discrete logarithm problems and will be very likely compromised in the post-quantum era.

5) BOUNDED ERROR QUANTUM POLYNOMIAL-TIME (BQP) PROBLEMS

The problems that can be solved by quantum computers in polynomial time. These include all the P problems and many NP problems (see Fig. 1), e.g., integer factorization (used in RSA cryptosystem) and discrete logarithm problem (used in DSA). Previous research indicates that most NP and all the NP-complete problems are outside the BQP zone [38] (see Fig. 1). Hence, a quantum computer needs more than a polynomial number of steps to solve an NP-complete problem.

NP-complete problems are ideal algorithmic problems for cryptography, particularly in the post-quantum era, because of their highest possible level of inherent difficulties (close to the NP class) when a given solution is verifiable in polynomial time. In other words, beyond that level of hardness, the verification of a solution cannot be performed in polynomial time which is not a desirable feature in cryptography. Moreover, they cannot be solved by quantum computers in polynomial time.

Since the security of code-based cryptosystems is based on the syndrome decoding (SD) problem (which is proved to be NP-complete [38], [39]), they can be suitable alternatives of conventional public-key cryptosystems. The next subsection briefly reviews the SD problem and discusses its NP-completeness.

B. SYNDROME DECODING (SD) PROBLEM

Before we review the SD problem, we briefly present some definitions of error-correcting codes.

Definition 1 (Block Code [30], [31]): The error correcting code $\mathbb{C}$ is a block code if the information bit streams are considered as separate fixed-length message segments (blocks).

In block codes, the information bit streams (i.e., input messages denoted by $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$) consist of k information digits defined over GF(q). The Galois field GF(q) is a field with a finite number of q distinct elements (e.g., GF(5) $= \{0, 1, 2, 3, 4\}$). A block code may comprise q^k possible distinct message blocks. The error correcting code $\mathbb{C}$ transforms each input message block $\boldsymbol{u}$ into a distinct n-bit ($n > k$) sequence denoted by $\boldsymbol{z}$. Each sequence $\boldsymbol{z}$ is called a codeword. As a result, q^k distinct legitimate codewords correspond to the q^k possible message blocks. The set of q^k codewords is referred to as a $\mathbb{C}(n, k)$ block code [40]. Unlike source coding mechanisms like MP4 (used for compressing data), in error correcting codes, $n - k$ redundant digits are purposefully added to the information block to attain error correction. Due to the added digits, an error correcting code incurs communication overhead accordingly.

Definition 2 (Rate of a Block Code [30], [31]): The rate (efficiency) of the block code $\mathbb{C}(n, k)$ is defined as k/n.

There is a trade-off between the communication overhead imposed by a block error correcting code and its effectiveness in correcting errors.

Definition 3 (Linear Block Code [30], [31]): The error correcting code $\mathbb{C}(n, k)$ is a linear code if the modulo-2 sum (i.e., the XOR binary operation if $q = 2$) of any two or multiple codewords is a valid codeword.

For simplicity, we consider binary block codes that the digits of $\boldsymbol{u}$ and $\boldsymbol{z}$ are defined over GF(2).

Definition 4 (Hamming Weight and Hamming Distance [40]): The number of non-zero bits in a codeword is called the Hamming weight. The Hamming distance $d(\boldsymbol{x}, \boldsymbol{y})$ between two codewords $\boldsymbol{x}$ and $\boldsymbol{y}$ is the number of bit positions where $\boldsymbol{x}$ and $\boldsymbol{y}$ differ.

Therefore, $d(\boldsymbol{x}, \boldsymbol{0})$ is the Hamming weight of $\boldsymbol{x}$, where $\boldsymbol{0}$ is the vector containing n bits of 0. The minimum distance d_{min} of a linear code $\mathbb{C}$ is the minimum Hamming distance between any two distinct codewords.

Definition 5 (Generator Matrix [30], [31]): A generator matrix $\boldsymbol{G}$ of a linear code $\mathbb{C}(n, k)$ is a $k \times n$ matrix that defines the one-to-one mapping between the k-bit message block $\boldsymbol{u}$ and the corresponding n-bit codeword $\boldsymbol{z}$, such that $\boldsymbol{z}_{1\times n} = \boldsymbol{u}_{1\times k} \cdot \boldsymbol{G}_{k\times n}$.

The encoder uses $\boldsymbol{G}$ to generate the distinct codewords associated with each message block $\boldsymbol{u}$.

Definition 6 (Systematic Code [41]): A linear code $\mathbb{C}(n, k)$ is systematic if its generator matrix can be written in the form of $\boldsymbol{G} = [\boldsymbol{I}_k | \boldsymbol{A}_{k\times(n-k)}]$, where $\boldsymbol{I}_k$ is a $k \times k$ identity matrix and $\boldsymbol{A}$ is the $k \times (n - k)$ coefficient matrix.

When $\boldsymbol{G}$ is specified in the systematic form, each n-bit codeword $\boldsymbol{z}$ generated by $\boldsymbol{G}$ can be split into two parts. The first k bits are equal to the corresponding message block $\boldsymbol{u}$, and the second part has the $(n - k)$ redundant parity-check bits.

Definition 7 (Parity-Check Matrix [30], [31]): The parity-check matrix $\boldsymbol{H}$ of a linear code $\mathbb{C}(n, k)$ is an $(n-k)\times n$ matrix that is orthogonal to all the codewords of $\mathbb{C}$, i.e., a codeword $\boldsymbol{z}$ is in $\mathbb{C}$ if and only if $\boldsymbol{H} \cdot \boldsymbol{z}^T = \boldsymbol{0}$.

If $\boldsymbol{G}$ is in the systematic form (i.e., $\boldsymbol{G} = [\boldsymbol{I}_k | \boldsymbol{A}_{k\times(n-k)}]$), then $\boldsymbol{H}$ is obtained by $\boldsymbol{H} = [\boldsymbol{A}^T | \boldsymbol{I}_{n-k}]$. $\boldsymbol{H}$ represents the linear relationships that the bits of a codeword $\boldsymbol{z}$ must be ordered as a codeword of $\mathbb{C}$. Thus, the decoder uses $\boldsymbol{H}$ to verify whether the received vector is a valid codeword of $\mathbb{C}$.

After reviewing the basic concepts of error correcting codes (see [30] and [31] for more details), in the following, we discuss the SD problem.

Definition 8 (Syndrome Decoding Problem): Given three integers n, k, and t (where $k < n$ and $t \leq n$), a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{2^{(n-k)\times n}}$, and a vector $\boldsymbol{Y} \in \mathbb{F}_{2^{(n-k)}}$, the syndrome decoding problem searches for a vector $\boldsymbol{e} \in \mathbb{F}_{2^n}$ of the Hamming weight $\leq t$ such that $\boldsymbol{H} \cdot \boldsymbol{e}^T = \boldsymbol{Y}$.

Berlekamp et al. proved that the SD problem is NP-complete if the parity-check matrix $\boldsymbol{H}$ is randomly chosen [39]. The NP-completeness establishes an essential security assumption required for code-based cryptosystems. It justifies why a code-based cryptosystem is a competitive alternative to the current public-key cryptosystems in the post-quantum era. Code-based cryptosystems are built based on the assumption that the efficient decoder of the code used in the encryption process is known only by the legitimate decrypting party (i.e., the party with the private key). Thus, any illegitimate decryption effort has to be done by solving the SD problem in which $\boldsymbol{H}$ is a randomly generated parity-check matrix. It makes the recovery of the error vector $\boldsymbol{e}$ (that is necessary for decryption) an NP-complete problem. Since this problem falls outside the BQP category of computational problems, code-based cryptosystems are potentially considered quantum-resistant [38].

III. THE McEliece (ME) CRYPTOSYSTEM

In this section, we review the ME cryptosystem as the basis of the CME scheme, analyze the security of ME and discuss its practical challenges.

A. DESCRIPTION

The ME cryptosystem is a public-key scheme invented by Robert McEliece in 1978 [25]. ME is based on error correcting codes, and its security relies on the NP-completeness of the SD problem. The main idea behind ME is to choose an error correcting code for which a fast and efficient decoding algorithm is known (McEliece suggested Goppa codes [42]). To encrypt a message, a disguised version of the generator matrix (of the selected code) is used as the public key to encode the input plaintext message. This setup forces any entity without the knowledge of the associated private key to use the syndrome decoding approach to decrypt the ciphertext, resulting in an NP-complete task according to the SD problem. However, the owner of the private key eliminates the disguise and decrypts the result using the known fast decoder.

The ME cryptosystem is composed of three algorithms: Key Generation (KeyGen), Encryption (*Encrypt*), and Decryption (Decrypt). To send a message to Bob, Alice uses Bob's public key, which is associated with his private key. The encryption of a message with Bob's public key transforms the message to a codeword (i.e., the ciphertext), decrypted upon reception by Bob using his private key.

Next, we briefly describe the KeyGen algorithm and elaborate on the steps involved in Encrypt and Decrypt algorithms.

1) KEY GENERATION

$(pk, sk) = KeyGen(m, t)$.

The KeyGen algorithm takes the integers m and t as input and returns the public-private key pair (pk, sk) by performing the following steps.

- To generate a key pair, Bob first generates a Goppa code $\mathbb{G}(n, k)$ that can correct t errors. This operation is accomplished by choosing a random Goppa polynomial $g(x)$ with degree t over Galois Field GF(2^m). This Goppa polynomial is defined as:

$$g(x) = g_0 + g_1 x + \cdots + g_t x^t = \sum_{i=0}^{t} g_i x^i \qquad (1)$$

  where $n = 2^m$ and $k = n - tm$ (According to the characteristics of Goppa codes. It should be noted that $g(x)$ is created through the random selection of g_i coefficients over GF(2^m), i.e., g_i $(i = 0, 1, \ldots, t)$ is a random binary number of m bits. See [42] for more details about Goppa codes).
- Based on the created Goppa code, Bob obtains its generator matrix $\boldsymbol{G}$ of dimension $k \times n$.
- Since $\boldsymbol{G}$ is used for transforming a plaintext message into a codeword (see Section II for details), Bob now scrambles this matrix by multiplying it with a $k \times k$ nonsingular matrix $\boldsymbol{S}$, and $n \times n$ permutation matrix $\boldsymbol{P}$ (i.e., to disguise $\boldsymbol{G}$). The resultant matrix $\boldsymbol{G}_{pub} = \boldsymbol{S} \cdot \boldsymbol{G} \cdot \boldsymbol{P}$ is published as the public key by Bob along with t, i.e., $pk = (\boldsymbol{G}_{pub}, t)$.
- Bob's private key is composed of $\boldsymbol{G}$, $\boldsymbol{S}$, and $\boldsymbol{P}$, i.e., $sk = (\boldsymbol{G}, \boldsymbol{S}, \boldsymbol{P})$.

2) ENCRYPTION

$\boldsymbol{c} = Encrypt(\boldsymbol{u}, pk)$.

To encrypt a k-bit plaintext message $\boldsymbol{u}$ with Bob's public key pk, Alice performs the following three steps (see Fig. 2).

- Alice first multiplies $\boldsymbol{u}$ with $\boldsymbol{G}_{pub}$ to get the n-bit vector $\boldsymbol{z} = \boldsymbol{u} \cdot \boldsymbol{G}_{pub}$.
- Then, she uses t (the other component of public key) and generates a random binary vector $\boldsymbol{e}$ of length n with Hamming weight t (see Section II for Hamming weight).
- Finally, she adds $\boldsymbol{z}$ and $\boldsymbol{e}$ to get the ciphertext $\boldsymbol{c} = \boldsymbol{z} \oplus \boldsymbol{e}$ before sending it to Bob (note that addition in module 2 is equivalent to a xor operation).

3) DECRYPTION

$\boldsymbol{u} = Decrypt(\boldsymbol{c}, sk)$.

To decrypt ciphertext $\boldsymbol{c}$, Bob uses his private key sk and performs the following steps:

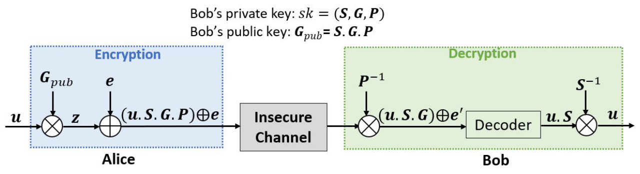

Bob's private key: $sk = (\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P})$
Bob's public key: $\boldsymbol{G}_{pub} = \boldsymbol{S}.\boldsymbol{G}.\boldsymbol{P}$
Encryption
$\boldsymbol{G}_{pub}$
$\boldsymbol{e}$
$\boldsymbol{u}$
$\boldsymbol{z}$
$(\boldsymbol{u}.\boldsymbol{S}.\boldsymbol{G}.\boldsymbol{P})\oplus\boldsymbol{e}$
Alice
Insecure Channel
Decryption
$\boldsymbol{P}^{-1}$
$(\boldsymbol{u}.\boldsymbol{S}.\boldsymbol{G})\oplus\boldsymbol{e}'$
Decoder
$\boldsymbol{u}.\boldsymbol{S}$
$\boldsymbol{S}^{-1}$
$\boldsymbol{u}$
Bob

FIGURE 2. Block diagram of the McEliece (ME) cryptosystem. Note that $\boldsymbol{e}' = \boldsymbol{e} \cdot \boldsymbol{P}^{-1}$ has the same hamming weight as $\boldsymbol{e}$. Thus, they both have the same effect on the decoder's functionality.

- Bob first multiplies ciphertext $\boldsymbol{c}$ with $\boldsymbol{P}^{-1}$ (i.e., a component of private key). It results in $\boldsymbol{c}' = \boldsymbol{u} \cdot \boldsymbol{S} \cdot \boldsymbol{G} \cdot \boldsymbol{P} \cdot \boldsymbol{P}^{-1} \oplus \boldsymbol{e} \cdot \boldsymbol{P}^{-1} = (\boldsymbol{u} \cdot \boldsymbol{S}) \cdot \boldsymbol{G} \oplus \boldsymbol{e}'$.
- $\boldsymbol{c}'$ appears to be a codeword of $\mathbb{G}$ (i.e., $(\boldsymbol{u} \cdot \boldsymbol{S}) \cdot \boldsymbol{G}$) that is corrupted by some errors $\boldsymbol{e}'$. To eliminate the error from this codeword, Bob uses the efficient Goppa decoder. The Hamming weight of $\boldsymbol{e}'$ and $\boldsymbol{e}$ are the same (i.e., t) because multiplying $\boldsymbol{e}$ with $\boldsymbol{P}^{-1}$ does not change the weight of $\boldsymbol{e}$ (it only scrambles its columns).
- Once the error is removed, the codeword $(\boldsymbol{u} \cdot \boldsymbol{S}).\boldsymbol{G}$ can be transformed into its corresponding message (i.e., $\boldsymbol{u} \cdot \boldsymbol{S}$). The generator matrix $\boldsymbol{G}$ is only known to Bob since this is part of his private key.
- Finally, to get the original message $\boldsymbol{u}$, Bob will simply multiply $\boldsymbol{u} \cdot \boldsymbol{S}$ with $\boldsymbol{S}^{-1}$, i.e., $\boldsymbol{u} \cdot \boldsymbol{S} \cdot \boldsymbol{S}^{-1} = \boldsymbol{u}.\boldsymbol{S}$ is known to Bob only since it is a part of his private key.

Fig. 2 shows the block diagram of the ME cryptosystem.

IV. VARIANTS OF McEliece CRYPTOSYSTEMS

The previous discussion shows that the original McEliece scheme is based on Goppa codes. Using Goppa codes for public key cryptosystems is widely studied and considered safe. However, the key sizes are an important problem with the original McEliece. Variants of the McEliece scheme have been proposed that target a reduction in key sizes [43]. One simple variant is proposed in [44] that does not need permutation or scrambling matrices as in the original McEliece scheme. Generally, these code-based variants allow iterative decoding like Low-Density Parity-Check (LDPC) codes. However, they are proven unsuitable for cryptosystems due to their sparse parity check matrix and the parity check equations corresponding to codewords in a dual code [43]. According to [44], a potential solution to this problem is to increase the weight of parity checks such that it is smaller than the dimension of the code, thereby making it cumbersome to find low-weight codewords in the dual code. This family of codes is called Moderate-Density Parity-Check (MDPC) codes that can be decoded with the same decoders as LDPC codes. For practical reasons, the quasi-cyclic version of MDPC (i.e., QC-MDPC) codes is particularly interesting because the parity check matrix is described by its first row only, resulting in a significant reduction in key sizes [43]. This section presents the details of QC-MDPC codes and discusses their Bit Flipping (BF) decoder. Then, the QC-MDPC variant of the McEliece scheme is presented, and its security issues are discussed.

A. QC-MDPC CODES

We present the Quasi-cyclic (QC) codes' definition as it is needed to understand QC-MDPC codes.

Definition 9 (Quasi-Cyclic Code): A binary linear code $\mathbb{C}(N, K)$ is quasi-cyclic if there exists an integer $n_0 < N$ such that every cyclic shift of a codeword $C \in \mathbb{C}$ by n_0 bits creates another valid codeword of $\mathbb{C}$.

In a systematic QC code, each codeword C consisted of p blocks of n_0 bits (i.e., $N = n_0 \times p$, where p is an integer), and every block includes $k_0 = K/p$ information bits and $r_0 = (n_0 - k_0)$ redundancy (parity) bits. For $r_0 = 1$, we have

$$r = (n_0 - k_0) \times p = r_0 \times p = p \tag{2}$$

In a QC code with $r_0 = 1$ (or $r = p$), the generator and parity check matrices comprise $p \times p$ circulant blocks [45]. The parity check matrix of a QC code can be written in the following format:

$$\boldsymbol{H} = [\boldsymbol{H}_0 \ \boldsymbol{H}_1 \ \ldots \ \boldsymbol{H}_{n_0-1}] \tag{3}$$

where each block $\boldsymbol{H}_i$ is a $p \times p$ binary matrix in the form of

$$\boldsymbol{H}_i = \begin{bmatrix} h_0^{(i)} & h_1^{(i)} & h_2^{(i)} & \ldots & h_{p-1}^{(i)} \\ h_{p-1}^{(i)} & h_0^{(i)} & h_1^{(i)} & \ldots & h_{p-2}^{(i)} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ h_1^{(i)} & h_2^{(i)} & h_3^{(i)} & \ldots & h_0^{(i)} \end{bmatrix}$$

Because $p = N - K$, $\boldsymbol{H}$ has $N - K$ rows and $n_0 p = N$ columns. As is evident, each block $\boldsymbol{H}_i$ can be described by its first row only, i.e., the other $p-1$ rows are obtained by cyclic shifts of the first row. Thus, to construct $\boldsymbol{H}$, one needs only the first row of n_0 $\boldsymbol{H}_i$ blocks. Since each row of $\boldsymbol{H}_i$ has p bits, only $n_0 p = N$ bits are needed to store $\boldsymbol{H}$.

Using the above $\boldsymbol{H}$, the generator matrix of a QC code for $r = p$ can be written in the following format:

$$\boldsymbol{G} = [\boldsymbol{I}_K | \boldsymbol{Q}_{K\times(N-K)}] = \left[\begin{array}{c|c} & (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_0)^T \\ & (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_1)^T \\ & (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_2)^T \\ \boldsymbol{I}_K & . \\ & . \\ & . \\ & (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_{n_0-2})^T \end{array}\right] \tag{4}$$

To prove the above format for $\boldsymbol{G}$, we use $\boldsymbol{H}\boldsymbol{G}^T = \boldsymbol{0}$ to obtain the following:

$$\begin{bmatrix} h_0^{(0)} & \ldots & h_{p-1}^{(0)} & \ldots & h_0^{(n_0-1)} & \ldots & h_{p-1}^{(n_0-1)} \\ h_{p-1}^{(0)} & \ldots & h_{p-2}^{(0)} & \ldots & h_{p-1}^{(n_0-1)} & \ldots & h_{p-2}^{(n_0-1)} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ h_1^{(0)} & \ldots & h_0^{(0)} & \ldots & h_1^{(n_0-1)} & \ldots & h_0^{(n_0-1)} \end{bmatrix} \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ 0 & 0 & \ldots & 1 \\ q_{0,0} & q_{2,0} & \ldots & q_{K,0} \\ q_{0,1} & q_{2,1} & \ldots & q_{K,1} \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ q_{0,(N-K)} & q_{1,(N-K)} & \ldots & q_{K,(N-K)} \end{bmatrix} = \boldsymbol{0}$$

The first K columns of $\boldsymbol{H}$ are $[\boldsymbol{H}_0 \ldots \boldsymbol{H}_{n_0-2}]$ because $(n_0 - 1)p = N - p = K$. The K columns are multiplied by $\boldsymbol{I}_K$ (the upper K rows of the second matrix) which results in

$$[\boldsymbol{H}_0 \, \boldsymbol{H}_1 \ \ldots \boldsymbol{H}_{n_0-2}] + \boldsymbol{H}_{n_0-1} \times \boldsymbol{Q}^T = \boldsymbol{0} \tag{5}$$

The aforementioned equation suggests that only the last block of $\boldsymbol{H}$ (i.e., $\boldsymbol{H}_{n_0-1}$) is multiplied by $\boldsymbol{Q}^T$ (the lower $N - K$ rows of the second matrix). $\boldsymbol{H}_{n_0-1}$ has $p = N - K$ columns which is equal to the number of rows in $\boldsymbol{Q}^T$. Since the above addition is in module 2, we can transfer the first term to the right side of the equation to obtain the following equation:

$$\boldsymbol{H}_{n_0-1} \times \boldsymbol{Q}^T = [\boldsymbol{H}_0 \, \boldsymbol{H}_1 \ \ldots \boldsymbol{H}_{n_0-2}] \tag{6}$$

This equation can be re-written as:

$$\boldsymbol{Q}^T = [(\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_0) \ \ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_1) \ \ \ldots \ \ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_{n_0-2})] \tag{7}$$

Finally, we obtain $\boldsymbol{Q}$ expressed as follows:

$$\boldsymbol{Q} = [(\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_0) \ \ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_1) \ \ \ldots \ \ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_{n_0-2})]^T = \begin{bmatrix} (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_0)^T \\ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_1)^T \\ . \\ . \\ . \\ (\boldsymbol{H}_{n_0-1}^{-1} \cdot \boldsymbol{H}_{n_0-2})^T \end{bmatrix}$$

which is identical to the format of $\boldsymbol{Q}$ in Eq. 4.

Definition 10 (LDPC/MDPC Codes): An (N, r, w)-LDPC or MDPC code is a linear code of length N and dimension $K = N - r$ that is defined by a parity-check matrix with a constant row weight w.

The only difference between LDPC and MDPC codes is in the value of row weight w. For LDPC codes, w is a small constant (usually less than 10); in MDPC codes, row weights are the function of code length, i.e., increase with code length N as $\mathcal{O}(\sqrt{NlogN})$. As discussed before, we are interested in MDPC codes only because the LDPC-based McEliece variants suffer from various vulnerabilities [43].

B. THE QC-MDPC VARIANT OF THE McEliece SCHEME

This variant of the McEliece cryptosystem was proposed by Misoczki et al. [44] in 2013. It consisted of three subroutines:

(1) Key Generation:

- **Private key**: is the parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{r \times N}$ of a t-correcting (N, r, w) code from QC-MDPC family that is defined as $\boldsymbol{H} = [\boldsymbol{H}_0\ \boldsymbol{H}_1\ \ldots\ \boldsymbol{H}_{n_0-1}]$. The first row of $\boldsymbol{H}$ is generated by selecting a random vector of $N = n_0 \times p$ bits. The other $r-1$ rows are obtained through cyclic shifts of the first row. Each block H_i in above equation is a circulant block of order $p \times p$, with $p = r$ and Hamming weight of w_i such that $w = \sum_{i=0}^{n_0-1} w_i$. The size of private key H is $p \times n_0 \times p = (N-k) \times N$. However, only N bits are needed to store the private key (the first row only).
- **Public key:** is the corresponding generator matrix $\boldsymbol{G}$ that can be computed from the private key (or $\boldsymbol{H}$) using Eq. 4. Similar to H, G is also quasi-cyclic that needs storing only the first row instead of the entire matrix. From the above matrix, it is evident that the size of $\boldsymbol{G}$ is $K \times (K + p) = K \times N$ (because $p = r = N - K$). As indicated above, the public key $\boldsymbol{G}$ can be described using its first row only, which has $K + p$ bits. Since the first K bits of the first row are part of the identity matrix $\boldsymbol{I}_K$, the actual size of the key is $p = N - K$. It significantly reduces the key size compared to $N \times K$ in the original McEliece scheme. For example, for the parameters $N = 1024$ and $K = 524$ as used in the original McEliece, the key size is drastically reduced from 66 KB to 63 B.

(2) Encryption:

The encryption of the QC-MDPC variant is akin to the original McEliece, in which a plaintext message $\boldsymbol{u} \in \mathbb{F}_2^K$ is subjected to the following computation:

$$\boldsymbol{r} = \boldsymbol{uG} + \boldsymbol{e} \tag{8}$$

where $\boldsymbol{e} \in \mathbb{F}_2^N$ is a random vector of $weight(\boldsymbol{e}) \leq t$, and the value of t is obtained from the error correcting capability of the corresponding decoder [44].

(3) Decryption:

To decrypt ciphertext $\boldsymbol{r} \in \mathbb{F}_2^N$, the receiver performs the following procedure:

- Compute $\boldsymbol{uG}$ from $\boldsymbol{r}$ by subjecting it to a t-error correcting decoder $\boldsymbol{\Psi}_H$ that leverages the knowledge of $\boldsymbol{H}$ for efficient decoding.
- Retrieve $\boldsymbol{u}$ from $\boldsymbol{uG}$. It can be accomplished by taking the first K bits of $\boldsymbol{uG}$ as $\boldsymbol{G}$ is in a systematic form.

In the QC-MDPC variant, there is no need to deploy the permutation and scrambling matrices $\boldsymbol{P}$ and $\boldsymbol{S}$. The simplification makes the decryption procedure simpler than the original McEliece scheme where $\boldsymbol{P}$ and $\boldsymbol{S}$ are critical elements (regardless of the complexity of the decoder unit that is discussed in the next section).

1) DEALING WITH CHOSEN CIPHERTEXT ATTACKS

Using the systematic format of the generator matrix $\boldsymbol{G}$ in the encryption procedure puts the scheme vulnerable to chosen ciphertext and message recovery attacks. It is because the ciphertext $\boldsymbol{r} = \boldsymbol{uG} + \boldsymbol{e}$ includes a copy of the plaintext $\boldsymbol{u}$ in its first K bits (since $r_i = u_i + e_i$ for $1 \leq i \leq K$). Two ciphertexts $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are most likely distinguishable if an attacker knows their corresponding plaintexts $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. In the worst case, if $e_i = 0$ for $1 \leq i \leq K$, the ciphertexts are distinguishable with the probability of 1.

To address this issue, Misoczki et al. [44] proposed to deploy a secure CCA2-conversion model in the QC-MDPC variant. Before encryption, the plaintext $\boldsymbol{u}$ is applied to a CCA2-conversion unit. It converts plaintext $\boldsymbol{u}$ to a random vector whose observation brings no useful knowledge for an adversary. Misoczki et al. recommended using the CCA2-Conversion model proposed by Kobara and Imai in [46].

C. DECODING QC-MDPC CODES

The proposed QC-MDPC variant of the McEliece scheme is based on the assumption that there is an efficient decoder $\boldsymbol{\Psi}_H$ to decode $\boldsymbol{uG} + \boldsymbol{e}$ to $\boldsymbol{uG}$. Misoczki et al. proposed the Bit Flipping (BF) algorithm to decode their QC-MDPC codes [44]. However, BF-based decoders are inherently probabilistic decoders that lead to a non-zero decoding failure probability. In such decoders, non-negligible decoding failure rates (DFR) degrade system efficiency and may result in critical security issues (presented in the next section). DFR is defined as the percentage of decoding failures in a given number of decoding attempts [44]. DFR is a critical factor in the efficiency and security of the QC-MDPC variant. In this regard, Misoczki et al. [44] proposed to select the system parameters to achieve a negligible DFR. In the following, we review the BF decoding algorithm and present its modified version proposed by Misoczki et al. for decoding QC-MDPC codes. Then, we investigate the DFR issue in detail and review the approaches proposed to deal with this issue.

1) BIT FLIPPING (BF) DECODER

The Bit Flipping algorithm was proposed by Gallager in 1963 [47]. It is a probabilistic and iterative decoding algorithm that provides an error correction capability that increases with the code length N and decreases with the weight of parity-check equations (i.e., w) [44]. As a quick result, the error correcting capability of BF in MDPC codes is lower than in LDPC codes (because LDPC codes have a smaller w).

The BF algorithm is defined based on the Tanner graph of an LDPC/MDPC code. The Tanner graph is a visual representation of the parity check matrix and has two sets of nodes:

- Check nodes (illustrated by squares in Fig. 3) represent the rows of parity-check matrix. Thus, every Tanner graph has $N - K$ check nodes (each representing a row of $\boldsymbol{H}$).
- Variable (Bit) nodes (illustrated by circles in Fig. 3) represent the N bits of a received vector $\boldsymbol{r}$ that should be decoded. Each bit node in the Tanner graph represents a single column of the parity-check matrix, as the size of r equals the number of columns in H.

The following procedure is performed to generate the Tanner graph of an LDPC/MDPC code with the $(N - K) \times N$ parity-check matrix $\boldsymbol{H}$ in which h_{ij} is the element located at row i and column j.

- Draw $N - K$ check nodes (squares) and N bit nodes (circles).
- Connect check node i to bit node j if $h_{ij} = 1$.

The number of edges that are connected to check node i (i.e., degree of the node) indicates the number of codeword bits that are involved in the ith parity-check equation (defined by row i of $\boldsymbol{H}$).

The BF decoding algorithm works as follows:

1) We generate the Tanner graph of the code using the parity-check matrix $\boldsymbol{H}$ (i.e., private key). We set $iter = 1$.
2) Given a specific vector $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ (that should be decoded), label each bit node j of the Tanner graph with r_j $(1 \leq j \leq N)$.
3) We compute the result of the parity-check equation for every row of $\boldsymbol{H}$, i.e., $\Delta_i = \sum_{j=1}^{N} h_{ij} r_j \bmod 2$, for $1 \leq i \leq N - K$.
4) If $\Delta_i = 0\ \forall i \in \{1, 2, \ldots, N - K\}$, we return $\boldsymbol{r}$ as the decoded vector and terminate the algorithm.
5) If $\Delta_i \neq 0$, we label the relevant check node i with "unsatisfied".
6) Make another label l_j for each bit node j, where l_j is the number of check nodes that (1) have a connection to bit node j, and (2) are labeled with "unsatisfied".
7) For each bit node j, if l_j is greater than a predetermined threshold b, then we flip r_j and update $\boldsymbol{r}$.
8) Set $iter = iter + 1$. If $iter$ is greater than a threshold and $\exists\ i \in \{1, 2, \ldots, N - K\}$ such that $\Delta_i \neq 0$, we return "failure" and terminate the algorithm.
9) With the updated $\boldsymbol{r}$, we jump to step 2.

Fig. 3 shows an example of how the BF algorithm decodes a received vector $\boldsymbol{r} = [0\,1\,0\,0\,1\,0\,0]$ that is corresponding with the codeword $\boldsymbol{z} = [0\,0\,0\,0\,0\,0\,0]$, i.e., $\boldsymbol{e} = [0\,1\,0\,0\,1\,0\,0]$.

Based on the presented procedure, the BF decoding algorithm is probabilistic. Decoding failures may occur because the maximum number of iterations may be reached before all the parity-check equations are satisfied. On the other hand, increasing the threshold determined for the number of iterations degrades the decoding performance in terms of speed. Specifically, the number of iterations is automatically greater for MDPC codes with a larger w than for LDPC codes. Intuitively, the complexity of the BF algorithm increases with w. To mitigate this issue, Misoczki et al. proposed modifying the original BF algorithm to select the threshold b. In the proposed approach, instead of a predetermined value of b, it is updated at each iteration using $b = MAX_u + \delta$, where MAX_u is the maximum number of unsatisfied parity-check equations at that iteration, and δ is a small integer. This approach is shown to be effective in reducing the overall number of iterations since it allows flipping a larger number of bits at each iteration [44]. Moreover, the algorithm can be restarted in case of a decoding failure by reducing δ by 1. In [44], the optimal initial value of δ is empirically determined as 5 to approximately reduce the number of iterations from 65 to 10.

2) MITIGATING THE DFR ISSUE

Although the new variant of the BF algorithm proposed by Misoczki et al. results in performance improvement of the BF-based decoder, the QC-MDPC variant still suffers from the DFR issue. The following three approaches have been proposed in [44] to mitigate the DFR issue.

- Approach 1: Conservative selection of the number of errors t and other system parameters such as N and w so that DFR becomes negligible (e.g., less than 10^{-7}).
- Approach 2: If the decoding process fails for a specific ciphertext, a more sophisticated decoding algorithm with better error correction capability is used. However, this approach increases the decoder's complexity.
- Approach 3: If the decoding process fails for a specific ciphertext, a new encryption will be requested (i.e., a retransmission request). Because a CCA2-conversion module is used, the two transmitted ciphertexts (associated with the encryption of the same message) are like random sequences (they are indistinguishable). Thus, an adversary cannot gain any information from this retransmission.

However, the last approach can make the scheme vulnerable to reaction attacks since the decoder informs the sender about the result of the last decoding procedure. A potential inception of such an attack is detailed below:

D. REACTION ATTACK ON QC-MDPC VARIANT

Employing a decoder with a non-negligible DFR may affect the QC-MDPC variant's security. Decoders with a noticeable DFR enable attackers to conduct a reaction attack. The first reaction attack was proposed by Guo et al. in 2016 [43]. Unlike other reaction attacks aiming to recover a single encrypted message, this attack targets recovering the private key in the QC-MDPC variant of the McEliece scheme with the BF decoder. Thus, it can be categorized as a structural attack as well. In this attack, it is assumed that $N = 2K$ and $n_0 = 2$. In this case, $\boldsymbol{H}$ is written in the form of $[\boldsymbol{H}_0\ \boldsymbol{H}_1]$. Moreover, it is assumed that $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$ have equal row weights, i.e., $w_0 = w_1$. Considering these assumptions, this attack targets to recover $\boldsymbol{H}_0$ (i.e., the first block of the private

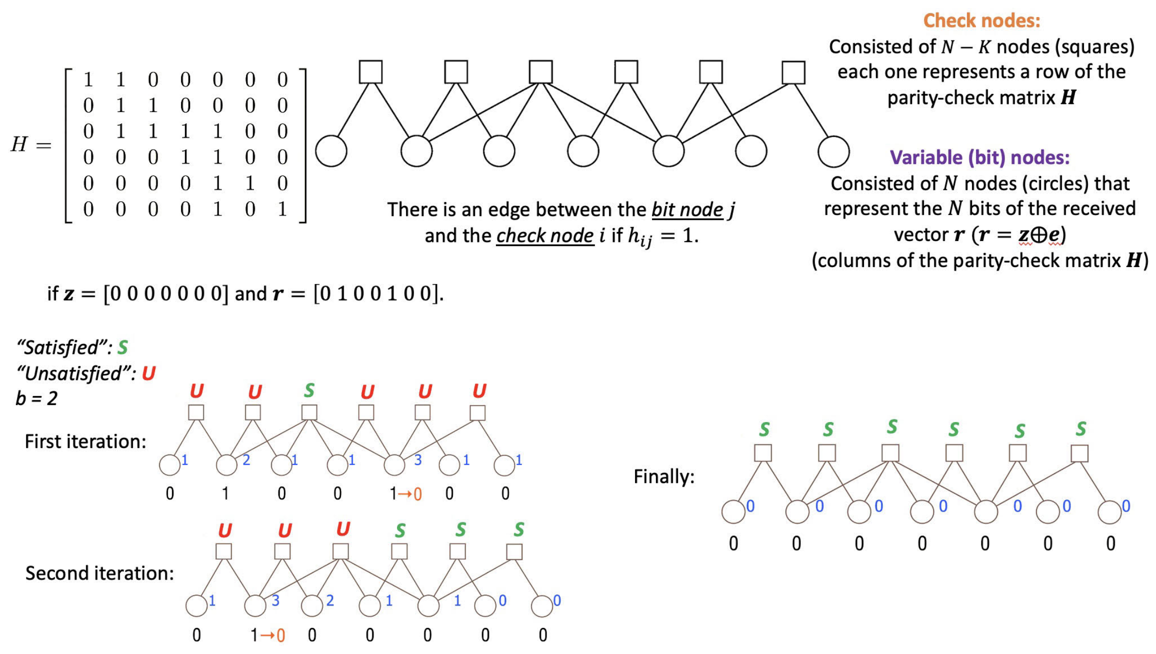

$H = \begin{bmatrix} 1&1&0&0&0&0&0\\ 0&1&1&0&0&0&0\\ 0&1&1&1&1&0&0\\ 0&0&0&1&1&0&0\\ 0&0&0&0&1&1&0\\ 0&0&0&0&1&0&1 \end{bmatrix}$
Check nodes:
Consisted of $N-K$ nodes (squares) each one represents a row of the parity-check matrix $\boldsymbol{H}$
Variable (bit) nodes:
Consisted of N nodes (circles) that represent the N bits of the received vector $\boldsymbol{r}$ ($\boldsymbol{r} = \boldsymbol{z} \oplus \boldsymbol{e}$)
(columns of the parity-check matrix $\boldsymbol{H}$)
There is an edge between the bit node j and the check node i if $h_{ij} = 1$.
if $\boldsymbol{z}$ = [0 0 0 0 0 0 0] and $\boldsymbol{r}$ = [0 1 0 0 1 0 0].
"Satisfied": S
"Unsatisfied": U
b = 2
First iteration:
U U S U U U
1 2 1 1 3 1 1
0 1 0 0 1→0 0 0
Second iteration:
U U U S S S
1 3 2 1 1 0 0
0 1→0 0 0 0 0 0
Finally:
S S S S S S
0 0 0 0 0 0 0
0 0 0 0 0 0 0

FIGURE 3. Tanner graph for a specific parity check matrix $\boldsymbol{H}$ and the BF decoding procedure on vector $\boldsymbol{r}$ = [0 1 0 0 1 0 0] that is corresponding with the codeword $\boldsymbol{z}$ = [0 0 0 0 0 0 0].

key) from $\boldsymbol{G}$ (i.e., public key) by sending a large number of encrypted messages and observing the reaction of the decoder at each decryption. Using $\boldsymbol{H}_0$, the attacker can obtain the remaining part of $\boldsymbol{H}$ (i.e., $\boldsymbol{H}_1$) by doing some linear algebra operations, such that $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$. In this regard, we have $[\boldsymbol{I}_K|\boldsymbol{Q}_{K\times K}][\boldsymbol{H}_0\ \boldsymbol{H}_1]^T = \boldsymbol{0}$ in which both $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$ are $K \times K$ (because for $N = 2K$, we have $p = N - K = K$). This results in $\boldsymbol{H}_0^T + \boldsymbol{Q}\boldsymbol{H}_1^T = \boldsymbol{0}$ since $\boldsymbol{I}_K\boldsymbol{H}_0^T = \boldsymbol{H}_0^T$. Finally, the attacker obtains $\boldsymbol{H}_1 = [\boldsymbol{Q}^{-1}\boldsymbol{H}_0^T]^T$.

To find $\boldsymbol{H}_0$, Guo et al. showed that there exists a strong correlation between the probability of decoding failure (for some chosen ciphertexts in which the error vector $\mathbf{e}$ has t 1s located at pairwise distance d) and the existence of a distance d between two 1s in the first row of $\boldsymbol{H}_0$. Obtaining the first row of $\boldsymbol{H}_0$ is sufficient for the attacker to recover the whole $\boldsymbol{H}_0$. In simple words, if there are two '1' digits in the first row of $\boldsymbol{H}_0$ at a distance d, then the probability of decoding failure is much smaller (in contrast with a situation when distance d does not exist between any two 1s.) Based on this finding, they first empirically computed the decoding failure probability for different values of d. Then, each d is classified into one of the two classes of "existing" and "not existing" based on the computed probability (i.e., a distance d with a small failure probability is categorized as "existing" and vice versa). This procedure can help the attacker reconstruct the first row of $\boldsymbol{H}_0$ for a successful private key recovery in minutes [43].

Similarly, the second approach may also create an opportunity for the attacker to obtain information about the result of decoding procedures through conducting side-channel attacks. For example, if approach 2 is adopted, an attacker can monitor the processing time taken by the decoder to decode a ciphertext and conclude whether the decoder had an unsuccessful decoding attempt (in the decoding of the previously-sent ciphertext). In case of failure, the decoder switches to another decoding technique, causing more latency than successful decoding. Thus, the first approach seems the most secure among the two. To fully address this decoding issue, the research community investigated the design of efficient decoders for QC-MDPC codes to achieve negligible DFRs [27], [48], [49].

Various alternative code-based cryptosystems are built on top of the aforementioned schemes, especially the ones included in the NIST's standardization project [17]. In the next section, we will discuss these state-of-the-art variants, including Classic McEliece (CM) [21] and Bit Flipping Key Encapsulation Scheme (BIKE) [20]). Both schemes are included in Round 3 of NIST's standardization project and are potential candidates for the post-quantum key encapsulation mechanism.

E. BIT FLIPPING KEY ENCAPSULATION SCHEME (BIKE)

BIKE [20] is another code-based KEM qualified in the third round of the NIST PQC standardization project as an alternate candidate and is based on QC-MDPC code. It leverages the Fujisaki-Okamoto (FO) CCA transform [50], [51] and the state-of-the-art Black-Gray-Flip (BGF) decoder [49] to address the IND-CCA insecurity issue of the QC-MDPC variant. Three different versions of the BIKE scheme exist, namely, BIKE-1, BIKE-2, and BIKE-3, that target the heterogeneous needs of different cryptographic applications. Following NIST's recommendation, BIKE has converged to a single version that relies heavily on BIKE-2. In the following, we review the final version of BIKE (i.e., spec v4.2 available in [20]) and succinctly describe its subroutines.

1) DEFINITIONS

We briefly present some definitions provided in the BIKE technical specification that are required to understand the subroutines of BIKE.

1) **System Parameters:** Based on the required level of security (denoted by λ), system parameters r, w, t, and l are determined.
   - r is the length of circulant blocks (equivalent to p as discussed earlier). Given $r = N/n_0$ and $n_0 = 2$ in BIKE, we have $N = 2r$. Moreover, since $r = N - K$, we have $N = 2r$ or $r = K$. r should be sufficiently large to result in (together with w and t) a low level of DFR such that the required security level λ is finally met (in BIKE, three security levels 1, 3, and 5 have been considered, corresponding to the security of AES-128, AES-192, and AES-256, respectively. To satisfy each level, a separate set of system parameters are recommended).
   - w is the row weight of the parity check matrix which is an even positive integer (i.e., $w/2$ is odd).
   - t is the Hamming weight of the error vector which is a positive integer.
   - l is the size of the generated (shared) symmetric key which is a positive integer.
2) **Hash Functions:** In BIKE, three hash functions $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{K}$ are uniformly selected at random that are modeled as random oracles. $\mathcal{H}$ takes an l-bit sequence and generates a $2r$-bit sequence of Hamming weight t (i.e., $\mathcal{H} : \{0,1\}^l \longrightarrow \{0,1\}^{2r}_{|t|}$). Similarly for $\mathcal{L}$ and $\mathcal{K}$ we have $\mathcal{L} : \{0,1\}^{2r} \longrightarrow \{0,1\}^l$ and $\mathcal{K} : \{0,1\}^{2l+r} \longrightarrow \{0,1\}^l$, respectively.

BIKE consists of three subroutines, key-generation, encapsulation, and decapsulation, depicted in Fig. 4. The ultimate result of these subroutines is a symmetric key secretly shared between two communicating parties. These subroutines work as follows:

2) KEY GENERATION SUBROUTINE

The KeyGen subroutine takes no input (except the system parameters) and generates a public-private key.

- **Private Key:** Generate $\boldsymbol{h}_0, \boldsymbol{h}_1 \longleftarrow \mathcal{R}$ both of weight $|\boldsymbol{h}_0| = |\boldsymbol{h}_1| = w/2$, where $\mathcal{R}$ is a cyclic polynomial ring $\mathbb{F}_2[X]/(X^r - 1)$ (equivalently, $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$ can be considered as $r \times 1$ column vectors). Then, select $\boldsymbol{\sigma}$ at random (uniformly) from the message space $\mathcal{M} = \{0,1\}^l$. Finally, set the private key as $\boldsymbol{sk} = (\boldsymbol{h}_0, \boldsymbol{h}_1, \boldsymbol{\sigma})$.
- **Public Key:** Compute $\boldsymbol{h} = \boldsymbol{h}_1\boldsymbol{h}_0^{-1}$ and send it to the other party as the public key $\boldsymbol{pk}$.

3) ENCAPSULATION SUBROUTINE

The *Encaps* subroutine takes the public key $\boldsymbol{h}$ as input and generates a ciphertext $\boldsymbol{C}$ (that is sent to the other party) and a symmetric key $\boldsymbol{K}_s$ that is remained secret and used for encrypting data. The following procedure is performed in this subroutine (see Fig. 4):

- Select an $l - bit$ vector $\boldsymbol{m}$ from the message space $\mathcal{M}$ uniformly at random.
- Compute $(\boldsymbol{e}_0, \boldsymbol{e}_1) = \mathcal{H}(\boldsymbol{m})$ where $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$ are error vectors of r bits such that $|\boldsymbol{e}_0| + |\boldsymbol{e}_1| = t$.
- Using the obtained error vectors $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$, compute $\boldsymbol{C} = (\boldsymbol{C}_0, \boldsymbol{C}_1) = (\boldsymbol{e}_0 + \boldsymbol{e}_1\boldsymbol{h}, \boldsymbol{m} \oplus \mathcal{L}(\boldsymbol{e}_0, \boldsymbol{e}_1))$ and send it to the other party.
- Compute $\boldsymbol{K}_s = \mathcal{K}(\boldsymbol{m}, \boldsymbol{C})$ as the secret symmetric key.

4) DECAPSULATION SUBROUTINE

The *Decaps* subroutine takes the private key $\boldsymbol{pk}$ and ciphertext $\boldsymbol{C}$ as input and generates the symmetric key $\boldsymbol{K}_s$ or a failure symbol $\perp$. The following procedure is performed in this subroutine:

- Compute the syndrome $\boldsymbol{S} = \boldsymbol{C}_0\boldsymbol{h}_0$.
- Decode $\boldsymbol{S}$ using the Black-Gray-Flip decoder (that will be reviewed in the next subsection) to obtain the error vectors $\boldsymbol{e}'_0$ and $\boldsymbol{e}'_1$. If $|\boldsymbol{e}'_0| + |\boldsymbol{e}'_1| \neq t$ or the decoding procedure fails, return $\perp$ and halt.
- Compute $\boldsymbol{m}' = \boldsymbol{C}_1 \oplus \mathcal{L}(\boldsymbol{e}'_0, \boldsymbol{e}'_1)$. If $\mathcal{H}(\boldsymbol{m}') \neq (\boldsymbol{e}'_0, \boldsymbol{e}'_1)$, set $\boldsymbol{m}' = \boldsymbol{\sigma}$.
- $\boldsymbol{K}_s = \mathcal{K}(\boldsymbol{m}', \boldsymbol{C})$.

Based on the above subroutines, two communicating parties can securely establish the same (symmetric) key $\boldsymbol{K}_s$. If the legitimate device (who owns the private key) receives a valid ciphertext $\boldsymbol{C}$, it will be able to compute $\boldsymbol{K}_s$ from $\boldsymbol{C}$. However, any malicious attempt to recover $\boldsymbol{K}_s$ without the knowledge of the private key fails because decoding the received ciphertext using syndrome $\boldsymbol{C}_0\boldsymbol{h}_0$ for an arbitrary QC code is infeasible (i.e., the SD problem is $\mathcal{NP}$-complete). Moreover, the legitimate receiver can verify the integrity of $\boldsymbol{C}_0$ (that is critical in recovering the shared key) through the confirmation of $\mathcal{H}(\boldsymbol{m}') = (\boldsymbol{e}'_0, \boldsymbol{e}'_1)$.

Although the submitters of BIKE provide specific implementation details [20], we present the essential aspects in the next subsection to avoid confusion.

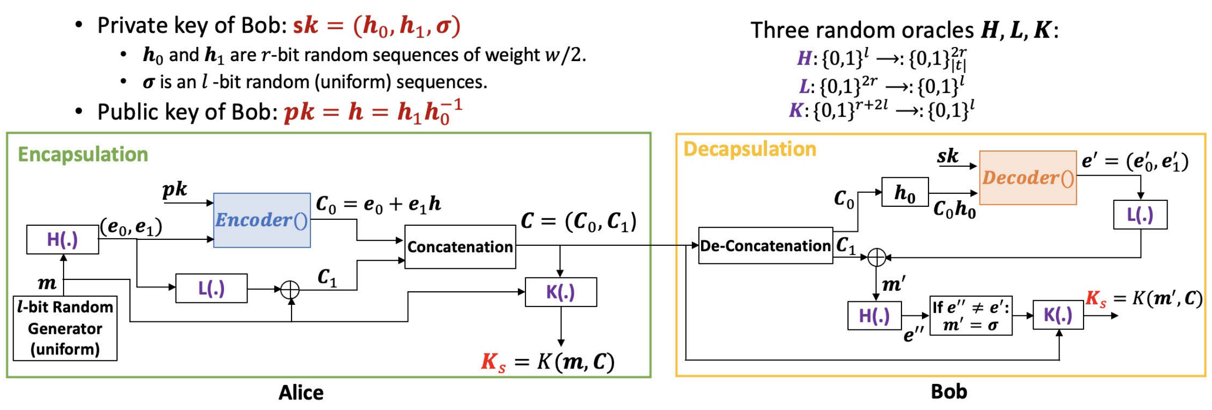
5) DETAILS ON BIKE

6) HOW IS THE ENCODING PROCEDURE PERFORMED IN THE *Encaps* SUBROUTINE?

The BIKE scheme is based on Niederreiter's framework [52] (like CM), so the encoding procedure is performed using the parity-check matrix. In BIKE, the parity-check matrix is deployed in its systematic format. Thus, for the encoding procedure, we have

$$\boldsymbol{C}_0 = \begin{bmatrix} \boldsymbol{e}_0 \ \boldsymbol{e}_1 \end{bmatrix} \cdot \boldsymbol{H}^T = \begin{bmatrix} \boldsymbol{e}_0 \ \boldsymbol{e}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{H}_1\boldsymbol{H}_0^{-1} \end{bmatrix}$$

Note that $\boldsymbol{H}^T$ is a $2r \times r$ matrix.



• Private key of Bob: $sk = (h_0, h_1, \sigma)$
• h_0 and h_1 are r-bit random sequences of weight $w/2$.
• σ is an l-bit random (uniform) sequences.
• Public key of Bob: $pk = h = h_1 h_0^{-1}$
Three random oracles H, L, K:
$H: \{0,1\}^l \longrightarrow: \{0,1\}^{2r}_{|t|}$
$L: \{0,1\}^{2r} \longrightarrow: \{0,1\}^l$
$K: \{0,1\}^{r+2l} \longrightarrow: \{0,1\}^l$
Encapsulation
pk
H(.)
(e_0, e_1)
Encoder()
$C_0 = e_0 + e_1 h$
Concatenation
$C = (C_0, C_1)$
m
l-bit Random Generator (uniform)
L(.)
C_1
K(.)
$K_s = K(m, C)$
Alice
Decapsulation
sk
Decoder()
$e' = (e'_0, e'_1)$
C_0
h_0
$C_0 h_0$
L(.)
De-Concatenation
C_1
m'
H(.)
e''
If $e'' \neq e'$: $m' = \sigma$
K(.)
$K_s = K(m', C)$
Bob

FIGURE 4. Block diagram of the BIKE scheme. Alice uses Bob's public key to securely generate a session key K_S. Once Bob receives the ciphertext C, he can obtain the same K_S as Alice generated (it should be mentioned that BIKE does not receive any plaintext as input because it is a KEM scheme used by two entities to secretly generate a shared key). After the shared session key generation, both parties can use K_S to communicate their message through a data encryption mechanism (DEM).

This is because $N = r+K = 2r$ (the dimension of $H_1 H_0^{-1}$ is $r \times r$). Thus, the final encoded vector is

$$C_0 = e_0 + e_1 H_1 H_0^{-1} \tag{9}$$

Equation 9 can be written in the following form since both H_0 and H_1 are circulant blocks that can be described using their first row.

$$C_0 = e_0 + e_1 h_1 h_0^{-1} = e_0 + e_1 h \tag{10}$$

7) WHY IS THE DECODING PROCEDURE PERFORMED USING $C_0 h_0$?

In the *Decaps* subroutine, the decoder module needs to recover (e_0, e_1) from the received ciphertext C_0. To do this, assume we have a QC code $\mathcal{C}$ with parity-check and generator matrices of $H = \begin{bmatrix} H_0 \ H_1 \end{bmatrix}$ and $G = \begin{bmatrix} H_1^T \ H_0^T \end{bmatrix}$, respectively ($\mathcal{C}$ is a valid code since the condition $GH^T = \mathbf{0}$ is satisfied for these matrices). Suppose that we have used $\mathcal{C}$ to encode the plaintext $u = [u_1 \ u_2 \ \ldots \ u_r]$ (note that $K = r$ in this code since $n_0 = 2$). Thus, the generated codeword $u.G$ is equal to $\begin{bmatrix} uH_1^T \ uH_0^T \end{bmatrix}$ (i.e., a $1 \times N$ vector, where $N = 2r$). Assume this codeword experiences an error vector e of length $2r$ and weight t that can be written in the form of $e = \begin{bmatrix} e_0 \ e_1 \end{bmatrix}$, where e_0 and e_1 are $1 \times r$ vectors such that $|e_0| + |e_1| = t$. In this case, if the result is vector r, we have the following:

$$r = u.G + e = \begin{bmatrix} uH_1^T + e_0 \ \ uH_0^T + e_1 \end{bmatrix} \tag{11}$$

Suppose we compute the syndrome of r. In this case, we will obtain the following by using the fact that the sum of the transpose of matrices is equal to the transpose of the sum of two matrices and by performing the simple matrix multiplication (note that if $M = [M_{ij}]$ is a matrix consisted of smaller blocks M_{ij}, then, $M^T = [M_{ji}^T]$).

$$\begin{aligned} S = H \cdot r^T &= \begin{bmatrix} H_0 \ H_1 \end{bmatrix} . \begin{bmatrix} (uH_1^T + e_0)^T \\ (uH_0^T + e_1)^T \end{bmatrix} \\ &= H_0((uH_1^T)^T + e_0^T) + H_1((uH_0^T)^T + e_1^T) \end{aligned} \tag{12}$$

Since H_0 and H_1 are circulant blocks, we have $H_0 H_1 = H_1 H_0$, $(uH_1^T)^T = H_1 u^T$, and $(uH_0^T)^T = H_0 u^T$. Thus, the syndrome S is written as follows:

$$\begin{aligned} S &= H_0 H_1 u^T + H_0 H_1 u^T + H_0 e_0^T + H_1 e_1^T \\ &= H_0 e_0^T + H_1 e_1^T \end{aligned} \tag{13}$$

This follows from the fact that addition is performed in module 2.

The above syndrome can be applied to a BF-based decoder (or any other appropriate decoder) to obtain e_0 and e_1. Now, we investigate the encoded vector $C_0 = e_0 + e_1 h_1 h_0^{-1}$ generated in the BIKE's *Encaps* subroutine. If we compute $C_0 h_0$, the result will be equivalent to the syndrome S, as shown in Eq. 13. Note that the circulant blocks H_0 and H_1 can be described using h_0 and h_1, respectively. h_0 and h_1 are considered as $r \times 1$ column vectors, i.e., H_0 and H_1 are constructed from h_0 and h_1 column-wise.

Therefore, we recover e_0 and e_1 from $S = C_0 h_0$ by applying it to a BF-based decoder that works using $H = \begin{bmatrix} H_0 \ H_1 \end{bmatrix}$. This is exactly what the decoder module deployed in *Decaps* does (note that the decoder takes h_0 and h_1 as input to decode $S = C_0 h_0$). However, since the BF decoder can lead to higher DFR, we discuss the Black-Gray-Flip (BGF) decoder recommended to be deployed in the *Decaps* subroutine of BIKE in the following subsection.

8) BGF DECODER

The BGF decoder [49] is a variant of the Black-Gray (BG) decoder [48], [53], which is a complex version of the BF decoder. The main difference between BG and the original BF decoder is that a BG decoder may flip back the value of a bit (variable) node (or an error bit, equivalently) if it is convinced that the bit was mistakenly flipped in the previous iteration. A simple version is the Time-to-Live (TTL) mechanism. However, in BG, a more complex version of TTL is used. In this version, the decoder creates two lists of bit nodes (called Black and Gray lists) to keep track of

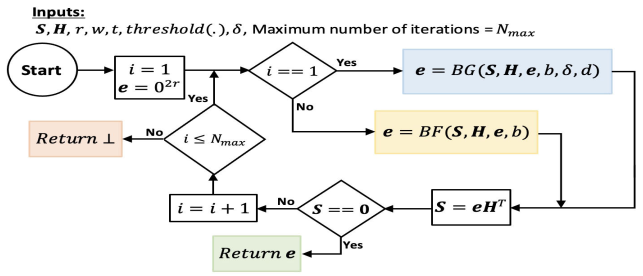


FIGURE 5. Black-Gray-Flip (BGF) decoder and its relation to Black-Gray (BG) and Bit Flip (BF) decoders.

those bit flips that are regarded as "uncertain" [54]. The first list (i.e., Black) maintains the positions of those bit nodes that were just flipped. However, the second list (i.e., Gray) keeps positions of bit nodes for which the number of unsatisfied parity checks is close to the threshold b but below it (i.e., greater than $b - \delta$, where δ is a small value chosen empirically).

In every iteration of BG, the following three steps are performed. In step 1, the decoder decides whether or not the value of a bit node should be flipped by comparing the corresponding unsatisfied parity checks with threshold b (akin to a single iteration of the BF decoder). In addition, it computes the black and gray lists. In steps 2 and 3, another iteration of BF is performed, but only the bit nodes in the black and gray lists are considered (for steps 2 and 3, respectively). In these steps, the value of black/gray bit nodes is updated if the number of their corresponding unsatisfied parity checks is greater than a newly set $threshold = [(d+1)/2] + 1$ (where $d = w/2$ is a positive odd integer, and the row weight w is a positive even integer), to gain more confidence in the flipped bits [49]. Finally, the value of the syndrome is updated based on the obtained error vector. If it is equal to zero, the obtained error vector is returned, and the algorithm terminates. Otherwise, if the current iteration number is less than the maximum number of iterations, another iteration is performed; otherwise, $\perp$ is returned, and the algorithm terminates.

The BGF decoder (deployed in BIKE), as depicted in Fig. 5, is similar to BG. It starts with one iteration of the BG decoder but proceeds with several BF iterations until either condition is met: The syndrome vector becomes zero, or the maximum number of iterations is reached (see Fig. 5). It translates to less number of required steps to achieve lower DFR as compared with the BG decoder. For example, for N_I iterations, BG would essentially need $3 \times N_I$ steps (N_I represents the number of iterations taken to recover all errors), whereas BGF needs $3 \times (N_I - 1)$ steps [49].

The BGF decoder has been identified as the most efficient variant of the BF algorithm in terms of DFR and complexity [49]. In BIKE, the thresholds used in BGF are computed using the functions tuned through extensive simulations. These threshold values are considered important system parameters since they determine the level of DFR in a decoder. In BIKE, the threshold values are obtained based on the target DFRs that are 2^{-128}, 2^{-192} and 2^{-256} for the respective levels of security (i.e., λ).

The formal proof of a negligible DFR is a perplexing task (no mathematical model has been proposed yet). Thus, in BIKE, the corresponding threshold values for different DFR levels are estimated through extensive simulations and extrapolations [55], [56].

9) BIKE SECURITY AGAINST REACTION ATTACKS

The QC-MDPC variant of the McEliece is vulnerable to a specific reaction attack that exploits the decoding failures to recover the private key [43]. This vulnerability has been seriously considered in BIKE since it deploys a QC-MDPC encoder. In this regard, BIKE submitters have proposed two approaches to address the reaction attack.

- Approach 1: This approach assumes that the party who sends the public key to initiate a session (i.e., the private key owner) generates a fresh public-private key pair for every session. In other words, this party never decapsulates more than one ciphertext with a single private key. Therefore, reaction-based key recovery attacks like the one proposed in [43] are no longer feasible because they need many observations of the decoder's reaction based on the same key pair. This approach is suitable for synchronous communication protocols (e.g., TLS) in which the immediate refreshing of the public-private key pair is possible. As a result, the mechanism needs to be IND-CPA secure only since CCA attacks are automatically addressed in such usage models. Moreover, this usage model provides both backward and forward secrecy.
- Approach 2: This approach is appropriate when a public-private key pair is reused for two or more symmetric key exchanges (i.e., two or more runs of the BIKE algorithm). It can occur either intentionally or inadvertently. For example, in asynchronous communication protocols (e.g., email), adoption of approach 1 is not feasible since the party that is sending data (through a DEM unit) needs a fresh version of the public key, which may not be provided yet by the other party (e.g., the other party is offline). In such scenarios, the deployed decoder must offer a negligible DFR, i.e., the system parameters r, w, and t must be carefully selected such that the DFR is low enough to match the security requirement.

The first approach results in a high level of latency due to the key generation procedure (and publishing of the updated public key) that needs to be performed at the beginning of each symmetric key exchange session. It can be a considerable issue for the key generation phase to take longer than the encapsulation procedure because it includes the computation of an inversion. According to the implementation results provided in the BIKE specification [20], the key generation procedure may take 2 to 4 times longer than the encapsulation procedure. Based on these results, the key generation phase shares 20 to 30 percent (approximately) in the overall latency. It indicates that updating the public-private key pairs for each key exchange session results in a 25 to 40 percent increase in latency. The large latency resulting from the decapsulation procedure partially mitigates this issue. Moreover, in the BIKE technical specification, deploying a batch key generation mechanism is proposed to allow computing only one polynomial inversion for every F key generation. Therefore, KeyGen is accelerated when establishing shared keys.

Moreover, reusing the key pair a few times does not result in an effective threat since the reaction-based key recovery attacks [43] rely on observing many decoding failures. Thus, adopting approach 2 with a non-negligible DFR seems secure if the number of key reuses is small.

10) BIKE SECURITY AGAINST SIDE-CHANNEL ATTACKS

In a side-channel attack, the attacker attempts to gain knowledge about a cryptosystem by observing the implementation-related parameters of the system (e.g., power consumption, timing information, etc.) rather than directly targeting the weaknesses that may exist in the system architecture [57]. In a specific class of side-channel attacks (timing attacks), the attacker targets to obtain knowledge by measuring the time taken to execute different algorithms, processes, or subroutines of the cryptosystem. This type of side-channel attack can be conducted on the QC-MDPC variant of the McEliece scheme by observing the time taken to decode different ciphertexts. Even if the decoder performs an implicit rejection when the decoding fails (such that the rejection brings no useful knowledge to an attacker), it is still possible for the attacker to identify decoding failures with high probability by measuring the decoding time. BF-based decoders have an iterative structure that causes the algorithm's run time to vary, i.e., the run time depends on the number of iterations required to find the decoded vector successfully. Thus, if it takes longer than expected for the decoding algorithm to terminate, the attacker (who monitors the run time) will know that a decoding failure has occurred.

In the BIKE specification, it is proposed to use a constant-time implementation of the BGF decoder to address side-channel attacks. In other words, the number of iterations is fixed for decoding different ciphertexts. Thus, based on the employed implicit-rejection version of Fujisaki-Okamoto transformation [50], [51], any malicious attempt to identify decoding failures will be unsuccessful. Using fresh public-private key pairs addresses the issue of reaction attacks (including side-channel attacks). However, as discussed in the previous subsection, this solution is not feasible for asynchronous applications. Thus, the constant-time implementation approach must be adopted to avoid side-channel attacks.

11) IMPACT OF WEAK-KEYS ON BIKE

Security of BIKE (IND-CCA security) is dependent upon the low DFR of the deployed decoder (e.g., BGF decoder as used in BIKE). For example, authors in [27] have argued that DFR needs to be as low as $2^{-\lambda}$ for λ-bit of IND-CCA security. However, interestingly, recent research has pointed out the existence of some specific (private) keys that negatively impact the DFR and the IND-CCA security. Such keys are referred to as weak-keys. For example, Drucker et al. [48] have argued that IND-CCA security of BIKE is difficult to claim without first proving (or disproving) the existence of weak-keys and formally quantifying their impact on DFR. Sendrier et al. [58] have demonstrated through empirical analysis that the average DFR is not hugely impacted by the known weak-keys. Thus, Drucker et al. argued that the existence of weak-keys is not critical to IND-CCA claims. However, this analysis is insufficient to claim the IND-CCA security of the BIKE. Subsequently, they conducted a similar analysis with the BGF decoder (see [59] for details) and argued that weak-keys do not significantly impact the DFR. However, another similar work [60] conducted a similar analysis with the BFG decoder and concluded different results compared with [59]. More precisely, this work (i.e., [60]) argued, based upon their analysis, that weak-keys could be a potential threat to IND-CCA security of BIKE. However, this difference could be using different parameters and platforms for their respective analysis. Our analysis suggests that some of the parameters in [59] are unclear, and subsequent empirical analysis with similar parameters may help ascertain weak-keys impact. Both these works are empirical, making it difficult to favor one or the other without understanding the root cause of the difference.

V. CONCLUSION AND FUTURE WORK

In this paper, we investigated the BIKE, i.e., a post-quantum key encapsulation scheme that has recently progressed to Round 4 of the NIST PQC standardization project and is likely to be standardized as a general-purpose KEM. This paper offers the necessary preliminaries and comprehends the orthodox ME cryptosystem and its contemporary variations based on QC-MDPC (e.g., BIKE) for fledgling researchers, security architects, and developers. We also comprehensively reviewed various possible attacks on such code-based cryptosystems and described the potential reasonings for the success or failure of such attacks. We comprehensively elaborated the working principles of the Bit-Flipping decoder and its state-of-the-art variants, such as the BGF decoder, and its role in facilitating certain types of reaction and side-channel attacks in BIKE. Our analysis of recent literature suggests that the weak-key issue has been described differently by different research works and demands a subsequent analysis by first benchmarking the approach, parameters, and potentially the data (e.g., ciphertexts and key-pairs used in DFR analysis). Although there have been no benchmarking efforts, we identify one dataset [61] for future analysis. Similarly, our review of related literature suggests no comparisons of BIKE with other candidates shortlisted in Round 4. We believe such a comparison may relieve interesting insights that may help in further tweaking this scheme. Similarly, making BIKE an authenticated-KEM (AKEM) is also currently overlooked and thus may open the door to some attacks that leverage the lack of authentication by design (see [62] for details). AKEM ensures that the party who initiates the symmetric key generation process (owner of the private key) can authenticate the other party when a ciphertext $\boldsymbol{C}$ is received to ensure the sender's legitimacy. It will help BIKE (or A-BIKE) to alleviate the possibility of various attacks.

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] F. PUB, "Digital signature standard (DSS)," FIPS PUB, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. FIPS PUB 186-4, 2000.
[3] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.
[4] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.
[5] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[6] S. Wang, Z. Cao, M. A. Strangio, and L. Wang, "Cryptanalysis and improvement of an elliptic curve Diffie–Hellman key agreement protocol," *IEEE Commun. Lett.*, vol. 12, no. 2, pp. 149–151, Feb. 2008.
[7] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Secur. Privacy*, vol. 16, no. 5, pp. 38–41, Sep. 2018.
[8] 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 of NIST Cyber Security White Paper (DRAFT) NIST CSWP 15, 2020, vol. 26.
[9] T. M. Fernández-Caramés, "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.
[10] *Quantum Computing and Post-Quantum Cryptography: Frequently Asked Questions*. Accessed: May 31, 2023. [Online]. Available: https://media.defense.gov/2021/Aug/04/2002821837/-1/-1/1/Quantum_FAQs_20210804.PDF
[11] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
[12] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219.
[13] R. A. Perlner and D. A. Cooper, "Quantum resistant public key cryptography: A survey," in *Proc. 8th Symp. Identity Trust Internet*. Gaithersburg, MD, United States: ACM, Apr. 2009, pp. 85–93.
[14] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1510–1523, Oct. 1997.
[15] G. Brassard, P. Høyer, and A. Tapp, "Quantum cryptanalysis of hash and claw-free functions," in *Proc. Latin Amer. Symp. Theor. Informat.* Cham, Switzerland: Springer, 1998, pp. 163–169.
[16] P. P. Pittalia, "A comparative study of hash algorithms in cryptography," *Int. J. Comput. Sci. Mobile Comput.*, vol. 8, no. 6, pp. 147–152, 2019.
[17] *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*, Standard NISTIR 8309, Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, 2020, doi: 10.6028/NIST.IR.8309.
[18] W. Beullens, Jan-Pieter, D'Anvers, A. Hülsing, T. Lange, L. Panny, C. de Saint Guilhem, and N. P. Smart, *Post-Quantum Cryptography*. Brussels, Belgium: European Union Agency for Cybersecurity, 2021, doi: 10.2824/92307.
[19] *PQC Standardization Process: Announcing Four Candidates to Be Standardized, Plus Fourth Round Candidates*. Accessed: May 31, 2023. [Online]. Available: https://csrc.nist.gov
[20] *Official Web Page of BIKE Suite*. Accessed: May 31, 2023. [Online]. Available: https://bikesuite.org
[21] *Official Web Page of Classic McEliece Suite*. Accessed: May 31, 2023. [Online]. Available: https://classic.mceliece.org
[22] *Official Web Page of HQC*. Accessed: May 31, 2023. [Online]. Available: https://pqc-hqc.org
[23] N. Sendrier, "Code-based cryptography: State of the art and perspectives," *IEEE Secur. Privacy*, vol. 15, no. 4, pp. 44–50, Aug. 2017.
[24] M. Repka and P.-L. Cayrel, "Cryptography based on error correcting codes: A survey," in *Multidisciplinary Perspectives in Cryptology and Information Security*, S. B. S. Al Maliky and N. A. Abbas, Eds. Hershey, PA, USA: IGI Global, 2014, pp. 133–156.
[25] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Coding Thv*, vol. 4244, pp. 114–116, Apr. 1978.
[26] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on post-quantum cryptography," U.S. Dept. Commerce, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 8105, 2016.
[27] N. Sendrier and V. Vasseur, "About low DFR for QC-MDPC decoding," in *Proc. 11th Int. Conf. Post-Quantum Cryptogr. (PQCrypto)*, vol. 12100. New York, NY, USA: Springer, 2020, pp. 20–34.
[28] E. Azougaghe, A. Farchane, I. Tazigh, and A. Azougaghe, *Comparative Study on the McEliece Public-Key Cryptosystem Based on Goppa and QCMDPC Codes*. Cham, Switzerland: Springer, 2021.
[29] H. Singh, "Code based cryptography: Classic McEliece," 2020, *arXiv:1907.12754*.
[30] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: Elsevier, 1977, vol. 16.
[31] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[32] D.-Z. Du and K.-I. Ko, *Theory of Computational Complexity*, vol. 58. Hoboken, NJ, USA: Wiley, 2011.
[33] O. Goldreich, "Computational complexity: A conceptual perspective," *ACM Sigact News*, vol. 39, pp. 35–39, Sep. 2008.
[34] A. Schrijver, "On the history of the shortest path problem," *Documenta Math.*, vol. 17, no. 1, pp. 155–167, 2012.
[35] C. Petit and J.-J. Quisquater, "Rubik's for cryptographers," *Cryptol. ePrint Arch.*, 2011.
[36] B. Barak, "The complexity of public-key cryptography," in *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. New York, NY, USA: Springer, 2017, pp. 45–77.
[37] L. Fortnow, "The status of the P versus NP problem," *Commun. ACM*, vol. 52, no. 9, pp. 78–86, Sep. 2009.
[38] S. Aaronson, "The limits of quantum," *Sci. Amer.*, vol. 298, no. 3, pp. 62–69, 2008.
[39] E. Berlekamp, R. McEliece, and H. 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.
[40] 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, Jun. 2011.
[41] 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. 1989–2016, 4th Quart., 2015.
[42] A. Valentijn, "Goppa codes and their use in the McEliece cryptosystems," Dept. Math., Syracuse Univ., Syracuse, NY, USA, Tech. Rep. 845, 2015.
[43] 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.* Cham, Switzerland: Springer, 2016, pp. 789–815.
[44] 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*, 2013, pp. 2069–2073.
[45] M. Baldi, *QC-LDPC Code-Based Cryptography*. Berlin, Germany: Springer, 2014.
[46] K. Kobara and H. Imai, "Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC," in *Proc. Int. Workshop Public Key Cryptography*. Cham, Switzerland: Springer, 2001, pp. 19–35.
[47] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[48] N. Drucker, S. Gueron, and D. Kostic, "On constant-time QC-MDPC decoding with negligible failure rate," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1289, Mar. 2019.
[49] N. Drucker, S. Gueron, D. Kostic, J. Ding, and J. Tillich, "QC-MDPC decoders with several shades of gray," in *Proc. PQCrypto*, vol. 12100, 2020, pp. 35–50.
[50] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki–Okamoto transformation," in *Proc. Theory Cryptography Conf.* Cham, Switzerland: Springer, 2017, pp. 341–371.

[51] 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. 364–374, Oct. 2021.
[52] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems Control Inf. Theory*, vol. 15, no. 2, pp. 159–166, 1986.
[53] 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.
[54] A. Nilsson, I. E. Bocharova, B. D. Kudryashov, and T. Johansson, "A weighted bit flipping decoder for QC-MDPC-based cryptosystems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1266–1271.
[55] N. Sendrier and V. Vasseur, "On the decoding failure rate of QC-MDPC bit-flipping decoders," in *Proc. Int. Conf. Post-Quantum Cryptography*. Cham, Switzerland: Springer, 2019, pp. 404–416.
[56] J. Tillich, "The decoding failure probability of MDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 941–945.
[57] V. Dragoi, T. Richmond, D. Bucerzan, and A. Legay, "Survey on cryptanalysis of code-based cryptography: From theoretical to physical attacks," in *Proc. 7th Int. Conf. Comput. Commun. Control (ICCCC)*, May 2018, pp. 215–223.
[58] N. Sendrier and V. Vasseur, "On the existence of weak keys for QC-MDPC decoding," 2020. [Online]. Available: https://eprint.iacr.org/2020/1232
[59] V. Vasseur, "QC-MDPC codes DFR and the IND-CCA security of BIKE," 2022. [Online]. Available: https://inria.hal.science/hal-03534003/
[60] M. R. Nosouhi, S. W. Shah, L. Pan, Y. Zolotavkin, A. Nanda, P. Gauravaram, and R. Doss, "Weak-key analysis for BIKE post-quantum key encapsulation mechanism," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2160–2174, 2023.
[61] M. Nosouhi, S. Shah, L. Pan, and R. Doss, "DU-QS22: A Dataset for analyzing QC-MDPC-based quantum-safe cryptosystems," in *Proc. Appl. Cryptography Comput. Commun. (AC3)*, 2022, pp. 3–10.
[62] *CWE-322: Key Exchange Without Entity Authentication*. Accessed: May 31, 2023. [Online]. Available: https://cwe.mitre.org/data/definitions/322.html

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 a Research Fellow with the Centre for Cyber Resilience and Trust (CREST), Deakin University, Australia. His research interests include post-quantum cryptography, next generation authentication systems, applied cryptography, and blockchain systems.

SYED WAJID ALI SHAH received the M.S. degree in electrical and electronics engineering from the University of Bradford, U.K., and the Ph.D. degree in computer science and engineering from the University of New South Wales (UNSW), Sydney, Australia. 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 "Advancing cyber security technologies" with CREST. 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 SECURITY COMPUTING, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. His research interests include cybersecurity and privacy.

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 system security, protocol design, and security analysis, with a focus on smart, cyber-physical, and critical infrastructures. His research program has been funded by the Australian Research Council (ARC) and the government agencies, such as the Defence 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 the Executive Council of the IoT Alliance Australia (IoTAA). He was a recipient of the Cyber Security Researcher of the Year Award from the Australian Information Security Association (AISA), in 2019.