Structural analysis of code-based algorithms of the NIST post-quantum call

M.A. GONZÁLEZ DE LA TORRE*, *Institute of Physical and Information Technologies (ITEFI) Spanish National Research Council (CSIC), Madrid, Spain.*

L. HERNÁNDEZ ENCINAS**, *Institute of Physical and Information Technologies (ITEFI) Spanish National Research Council (CSIC), Madrid, Spain.*

J.I. SÁNCHEZ GARCÍA†, *Institute of Physical and Information Technologies (ITEFI) Spanish National Research Council (CSIC), Madrid, Spain.*

Abstract

Code-based cryptography is currently the second most promising post-quantum mathematical tool for quantum-resistant algorithms. Since in 2022 the first post-quantum standard Key Encapsulation Mechanism, Kyber (a latticed-based algorithm), was selected to be established as standard, and after that the National Institute of Standards and Technology post-quantum standardization call focused in code-based cryptosystems. Three of the four candidates that remain in the fourth round are code-based algorithms. In fact, the only non-code-based algorithm (SIKE) is now considered vulnerable. Due to this landscape, it is crucial to update previous results about these algorithms and their functioning. The Fujisaki-Okamoto transformation is a key part of the study of post-quantum algorithms and in this work we focus our analysis on Classic McEliece, BIKE and HQC proposals, and how they apply this transformation to obtain IND-CCA semantic security. Since after security the most important parameter in the evaluation of the algorithms is performance, we have compared the performance of the code-based algorithms of the NIST call considering the same architecture for all of them.

Keywords: code-based cryptography, Fujisaki-Okamoto transformation, key encapsulation mechanism, post-quantum cryptography, public key encryption

1 Introduction

Post-quantum cryptography has increased in relevance in the past years and it is expected to keep growing in the near future. One of the main references in post-quantum cryptography advances is the National Institute of Standards and Technology (NIST) Post-Quantum (PQ) standardization process [12]. This process started after Shor published two quantum algorithms [14] allowing quantum computers to break the asymmetric cryptosystems whose security is based on the integer factorization or discrete logarithm problems. The NIST call started in 2016 and currently, in 2023, it is in the fourth round of evaluation. In summer 2022 CRISTALS-Kyber was announced as the first candidate to be standardized in the Public Key Encryption/Key Encapsulation Mechanism (PKE/KEM) cathegory. From the beginning, the NIST announced that it would like to establish more than one standard, each based on a different mathematical tool. Apart from lattices, there were other four different tools with quantum-resistant problems. Nevertheless, today only two of them remind:

*E-mail: ma.gonzalez@csic.es
**E-mail: luis.h.encinas@csic.es
†E-mail: ji.sanchez@csic.es

Vol. 00, No. 0, © The Author(s) 2024. Published by Oxford University Press.

https://doi.org/10.1093/jigpal/jzae071

lattices and hash functions. The third tool, error-correcting codes, is still under scrutiny once the fourth round has started.

The Fujisaki-Okamoto (FO) transformation is a general structure that transforms a sufficiently secure PKE to a highly secure KEM. Since the first introduction of this transformation [6], there are several generalizations and in-depth studies about it. From the initial conditions to the security reduction, all the aspects of this transformation play a key role in the final conception of a cryptosystem. In Post-Quantum Cryptography (PQC) this transformation is used for all the PKE/KEM candidates of the NIST call, therefore studying it gives a good comprehension of the algorithms.

This work is the expansion of a previous study presented to the *15th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2022)* [7]. It also completes our study of the FO transformation, along with publication [8], which looks at how the transformation applies to PQ lattice-based algorithms. The initial submission studied the version of the FO transformation applied to the code-based algorithms of the third round of the NIST call. These algorithms, Classic McEliece, BIKE and HQC, are currently under new study in the fourth round. Some changes were introduced in the fourth round submissions, related with the application of the FO transformation, which we analyze in this work. Also, a performance analysis of the latest version of the algorithms presented to NIST is included. This analysis allows to study the second criteria established by NIST for the evaluation of the algorithms, preferring short computational times and short sizes of keys and ciphertexts.

The rest of this paper is organized as follows. In Section 2 the theoretical background on code-based cryptography is introduced. In Section 3 we study how the different variants of the FO transformation are used in the code-based algorithms of the NIST call. Section 4 includes our results on the performance of the algorithms. Lastly, in Section 5 we present our conclusions for this study.

2 Theoretical background

2.1 Security definitions

To understand the FO transformation it is essential to remain aware of the different security notions involved. This transformation induces what is called a security reduction, meaning that the resulting KEM is, for a certain security notion, at least as hard to break as a bound that factors in the hardness of the underlying PKE. There are two main security definitions relevant in the study of the FO-transformation: One-Way (OW) and Indistinguishability (IND) security. OW security consists in the difficulty of finding the plaintext that corresponds to a certain ciphertext. The simplest definition of OW-security is called *One Way Chosen Plaintext Attacks* (OW-CPA) and the attack consists in the definition that was just provided above. Other definitions of OW-security notions consider specific oracles that the attacker can use, but those are not involved in the transformations applied to code-based algorithms, hence these definitions fall out of our range of study. These security definitions can be consulted in [8].

The strong security notions widely considered are the ones called *INDistinguishability under Chosen Plaintext Attack* (IND-CPA) and *INDistinguishability under Adaptive Chosen Ciphertext Attack* (IND-CCA). These notions of security also define an adversary with a certain amount of resources and the security of the cryptosystem consists in the inverse of the probability of success of this adversary. The difference between the OW- and the IND-security is in the game proposed to the adversary. So, given two plaintexts and a ciphertext, the adversary is asked to choose, between the plaintexts, which one corresponds to the given ciphertext. During the process of generating the plaintexts and the ciphertext, and the choosing process, the attacker has access to oracles, being the number of guesses another parameter. The definition of IND-CCA security consists in defining an

IND-CPA adversary that also has access to the decryption oracle (this oracle can not decrypt the challenged ciphertext).

When a security reduction is considered, it is desirable that the reduction is tight (see [13]). In the context of the FO transformation this means that for any attacker A against KEM security running at time t, there exists an attacker B against PKE security running at time t' such that $\mathrm{Adv}_{\mathrm{KEM}}(\mathrm{A}) \leq \mathrm{Adv}_{\mathrm{PKE}}(\mathrm{B})$, where $\mathrm{Adv}_{\mathrm{ALG}}(\mathrm{A})$ denotes the advantage of an attacker, A, against ALG in terms of the probability of success, and $t \approx t'$.

In this work, we consider that a Probabilistic PKE (PPKE) is a set $\pi = \{\mathcal{G}', \mathcal{E}, \mathcal{D}, M, R\}$, where $\mathcal{G}'$, $\mathcal{E}$ and $\mathcal{D}$ are the key generation, the encryption and the decryption algorithms, respectively. The set M is the set of possible messages and R is an optional randomness set. If the PKE is Deterministic (DPKE), then R is not considered. M can be omitted, to reduce the notation, in case it is not necessary to specify it. Moreover, we denote by $\kappa = \{\mathcal{G}, \mathcal{E}c, \mathcal{D}c\}$ a generic KEM, where $\mathcal{G}$, $\mathcal{E}c$ and $\mathcal{D}c$ are the key generation, the encapsulation and the decapsulation algorithms, respectively. The *correctness* of a PKE is related to the probability of generating invalid ciphertexts, i.e., ciphertexts generated by the encryption algorithm so that, if the decryption algorithm takes them as input, then it outputs an error: $\perp$. A PKE is said to be *perfectly correct* if for any pair of public-private keys, (pk, sk), generated by $\mathcal{G}'$, and for any $m \in M$, it is fulfilled that $\Pr[\mathcal{D}(sk, c) = m \colon c = \mathcal{E}(pk, m)] = 1$.

2.2 Code-based encryption

The theory *of error-correcting codes* is a mathematical speciality that allows to detect and to correct the possible errors produced when any information is transmitted. The errors can be due to noise, alteration, etc. This mathematical tool permits the receiver to recover the original information even if some of its bits have been erased or modified. This is done by adding redundant information, sometimes called *parity check*, by using matrix algebra over finite fields, $\mathbb{F}_p$.

In the encoding/decoding process only certain strings of bits (called *codewords*) are valid ciphertexts. A distance defined between valid codewords in the code space allows the user to estimate if a received codeword is valid or might be corrected to a valid one. The most commonly used distance is the Hamming distance, i.e., the number of different bits between two bit strings. In this work, the plaintexts are presented as binary strings, the binary field is denoted as $\mathbb{F}_2$, and the cyclic polynomial ring defined for $n \in \mathbb{N}$ is denoted as $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$.

3 FO transformation application in code-based algorithms

All the finalist and alternative post-quantum encryption schemes in the fourth round of the NIST call in the PKE/KEM chapter use the FO transformation. So, we will study how this transformation and some modifications of it are used in the code-based proposals. All the three proposals presented in this section use hash functions as cryptographic primitives, which will be denoted as H_i in the following algorithms, where i is an index that differentiates among the different hash functions used in a single proposal. In order to facilitate the understanding of the algorithms and functions, for each proposal we will keep its original notation.

3.1 Classic McEliece

Classic McElice [2] is a proposal based in the first code-based public-key cryptosystem, introduced by McEliece in [11]. Its security depends on binary Goppa codes; in particular, the public key is a random binary Goppa code. A ciphertext consists in a codeword with errors and the decoding process eliminates these errors. Classic McEliece is the only algorithm studied in this work that was a finalist in both the third and fourth (and current) rounds of the NIST call. It seems like a very

promising candidate and some of the changes introduced may slightly improve its performance. The parameters considered are the following:

- A positive integer m, that also defines a parameter $q = 2^m$.
- Two positive integers n, t, so that $n \leq q$ and $t \geq 2$ with $m \cdot t < n$. This also defines the parameter $k = n - m \cdot t$.
- A monic irreducible polynomial $f(z) \in \mathbb{F}_2[z]$ of degree m. This defines a representation $\mathbb{F}_2[z]/f(z)$ of the field $\mathbb{F}_q$.
- A monic irreducible polynomial $F(y) \in \mathbb{F}_q[y]$ of degree t. This defines a representation $\mathbb{F}_q[y]/F(y)$ of the field $\mathbb{F}_{q^t} = \mathbb{F}_{2^{mt}}$.
- Integers $\nu \geq \mu \geq 0$ with $\nu \leq k + \mu$. Both set to 0 if not specified otherwise.

In McEliece's submission there are a total of ten different parameter sets, which are a high amount in comparison with other PQ-algorithms. Some of these parameter sets are quite similar; in fact, half of the parameter sets are defined with $\nu = \mu = 0$ and the other half only change $\nu = 64$ and $\mu = 32$. For example kem/mceliece348864 is defined as: $m = 12$, $n = 3488$, $t = 64$, $f(z) = z^{12} + z^3 + 1$, $F(y) = y^{64} + y^3 + y + 1$ and $\nu = \mu = 0$. The parameter set kem/mceliece348864f is defined identically except for $\nu = 64$, $\mu = 32$. The security levels that each parameter set targets are the following:

Level 1: kem/mceliece348864 and kem/mceliece348864f.

Level 3: kem/mceliece460896 and kem/mceliece460896f.

Level 5: kem/mceliece6688128, kem/mceliece6688128f, kem/mceliece6960119, kem/mceliece6960119f, kem/mceliece8192128 and kem/mceliece8192128f.

Since McEliece PKE [2, §2.2] is deterministic (and also perfectly correct) the FO transformation is a reduced version of the original one, denoted as $\mathrm{U}^{\not\perp}$, which is used with implicit rejection and avoids the use of the encryption algorithm in the decapsulation. Moreover, the shared secret is defined as $H(1, e, C)$, where e and C are the plaintext and the ciphertext, respectively. Its IND-CCA security tightly reduces to the OW-CPA security of the underlying PKE in the Random Oracle Model (ROM) [4, Th.14.3]. The transformation is modified with the intention of applying both Dent's and Saito et al.'s [13, Th 5.2] results, which, originally, take into consideration different KEM constructions. The submission of Classic McEliece to the NIST states that there is no proper security reduction theorem in the Quantum ROM (QROM) for the transformation applied. However, the results from Saito et al. [13] can be applied to support the security of this proposal. Algorithm 1 shows how the $\mathrm{U}^{\not\perp}$ transformation is applied.

Classic McEliece KEM proposal executes the PKE algorithms: the Key Generation algorithm of the PKE is originally called SeedKeyGen, while the encryption and decryption algorithms are called Encode and Decode, respectively. Other functions used are: FixedWeight that generates a vector $e \in \mathbb{F}_2^n$ of weight t, and a hash function H. Also, the symbol Γ describes a binary Goppa code of length n and dimension k, which is part of the secret key and is used to decode the encrypted shared secret.

The transformation that McEliece applies has changed from the third round. Previously, an *additional hash* was used, i.e., the hash of the message e was added to the ciphertext and was later used in the decapsulation. The first QROM security reductions were introduced in [9] using such additional hash; however, in newer publications [10, 13], security reductions are introduced in the QROM that do not need the additional hash, and those are the most widely used among the PQ-algorithms.

3.2 BIKE

BIKE (*Bit flipping Key Encapsulation*) is a KEM based on Quasi-Cyclic Moderate Density Parity-Check (QC-MDPC) codes [3]. In the third round of the NIST call, BIKE was considered as an alternative instead of as a finalist, and now, in the fourth round, BIKE is included, offering a wider

Algorithm 1 McEliece KEM KeyGeneration, Encapsulation and Decapsulation

McEliece KEM KeyGeneration

$\delta \leftarrow \{0,1\}^l$
$(pk, sk) \leftarrow \text{SeedKeyGen}(\delta)$ [PKE.KeyGen]
return (pk', sk)

McEliece KEM Encapsulation(pk)

$e \leftarrow \text{FixedWeight}()$
$C = \text{Encode}(e, T)$ [PKE.Encrypt]
$K := H(1, e, C)$ [Compute K]
return (C, K)

McEliece KEM Decapsulation(sk, C)

$b \leftarrow 1$
Extract from sk: $s \in \mathbb{F}_2^n$ and $\Gamma' = (g, \alpha'_0, \ldots, \alpha'_{n-1})$
$e = \text{Decode}(C, \Gamma')$ [PKE.Decrypt]
In case of $e = \perp$, set $e = s$ and $b = 0$
$K := H(b, e, C)$ [Compute K]
return (K)

TABLE 1. BIKE parameter sets

Security level	r	w	t	key length (l)
1	12 323	142	134	256
3	24 659	206	199	256
5	40 973	274	264	256

range of code-based possibilities. The underlying PKE scheme is claimed to reach IND-CPA security (confirmed in [5]). Then the $\text{FO}^{\not\perp}$ transformation is applied to reach IND-CCA security, however the security reduction depends heavily on the δ-correctness of the decoder. In the submission to NIST, a sufficiently secure decoder is presented.

Table 1 shows the parameters that define BIKE: The block size r (the code length $n = 2r$), the row weight $w \approx \sqrt{n}$ (w even and $w/2$ odd) and the error weight $t \approx \sqrt{n}$. Based on these parameters, BIKE submission presents three parameter sets, targeting the levels 1, 3 and 5 of security established by NIST.

Due to the possibility of not reaching IND-CCA security, in BIKE submission it is suggested to use ephemeral keys, i.e., to consider a fresh public/private key pair for every key exchange session, for which an IND-CPA security is sufficient (models using the same key pair more than once require IND-CCA security).

In BIKE's submission it is claimed that it applies the $\text{FO}^{\not\perp}$ transformation, which is defined by two basic transformations T and $\text{U}^{\not\perp}$. Nevertheless, it is not formally defined an underlying PKE to which the transformation be applied. For the sake of similarity with the other code-based proposals, we have considered the following functions: KeyGenPr, EncryptPr(pk, e) and DecryptPr(sk, C) (see Algorithm 2). These functions are not defined in BIKE; however, we defined them as the algorithms that comprehend a PKE π such that $\text{FO}^{\not\perp}[\pi] = \text{BIKE}$. In [5] an underlying PKE for BIKE reversing the $\text{FO}^{\not\perp}$ that perfectly fits this definition is considered. Moreover, it is defined the set $\mathcal{H}_\omega = \{(h_0, h_1) \in \mathcal{R} \times \mathcal{R} : |h_0| = |h_1| = w/2\}$, where w is one of the parameters [3].

TABLE 2. HQC parameter sets

Security level	n_1	n_2	n	w	$w_r = w_e$
1	46	384	17 669	66	75
3	56	640	35 851	100	114
5	90	640	57 637	131	149

Initially, in the KeyGeneration algorithm, as part of the $U^{\not\perp}$ transformation, it is sampled a pseudo-random message σ. During the encapsulation, the randomness that is used in the coding process (EncryptPr) is generated by using the hash of the chosen e. This is part of the T transformation and makes BIKE deterministic. Also, in the encapsulation, the shared secret $K := H_1(m, c)$ is defined depending on both, the ciphertext and the plaintext. Finally, in the decapsulation it is only checked if $e'(m) = H_2(m')$. This process does not suppose a re-encryption and it belongs to the transformation $U^{\not\perp}$.

Algorithm 2 BIKE KEM KeyGeneration, Encapsulation and Decapsulation

BIKE KEM KeyGeneration

$h_0, h_1, h_1 \cdot h_0^{-1} \leftarrow$ KeyGenPr() [PKE.KeyGen]
$\sigma \leftarrow M$ [PrimitivesKG]
return $(pk := h,\ sk := (h_0, h_1, \sigma))$

BIKE KEM Encapsulation(pk)

$m \leftarrow \{0,1\}^l$ [PrimitivesEncaps]
$c \leftarrow$ EncryptPr(pk, m) [PKE.Encrypt]
$K := H_1(m, c)$ [Define K]
return (K, c)

BIKE KEM Decapsulation(sk, C)

Parse $c = (c_0, c_1)$ and $sk = (h_0, h_1, \sigma)$ [Fix inputs]
$m' \leftarrow$ DecryptPr(c, sk) [PKE.Decrypt]
If $e' = H_2(m')$, if true then $K = H_1(m', c)$, if not $K = H_1(\sigma, c)$ [Checking]
return (K)

3.3 HQC

HQC (*Hamming Quasi-Cyclic*) is an efficient encryption scheme also based on coding theory, whose PKE associated is IND-CPA secure [1,Th.5.1]. HQC applies the Quantum-ROM, $QFO^{\perp}$, transformation and it is stated that it reaches IND-CCA security in the ROM [9]. However, in [9], security proofs are presented only for the transformations $QFO_m^{\perp}$ and $QFO_m^{\not\perp}$. HQC submission also claims the transformation $FO^{\not\perp}$ could be applied to make the proposal IND-CCA secure in the QROM [4, 9, 10, 13].

Table 2 shows the HQC parameters for the three levels considered: n_1 and n_2 are the lengths of the codes used in the algorithm, n is the length of the ambient space and it is chosen as the smallest primitive prime greater than $n_1 n_2$. The weights of the vectors involved in the PKE algorithm are w, w_r and w_e.

Because in case of failure in the Decaps algorithm the output is $\perp$ (transformation with explicit rejection), it is not necessary to modify the KeyGen algorithm, hence it is the same in both HQC PKE and HQC KEM. In our study the running time considered was the of the PKE Key Generation algorithm.

Algorithm 3 HQC KEM KeyGen, Encapsulation and Decapsulation

HQC KEM KeyGen

$pk, sk \leftarrow$ HQC.PKE.KeyGen() [PKE.KeyGen]
return (pk, sk)

HQC KEM Encapsulation(pk)

Set $m \leftarrow \mathbb{F}_n^k, salt \leftarrow \mathbb{F}_2^{128}$ and compute $\theta \leftarrow G(e, seed2, salt)$ [Primitives Encaps]
Parse $pk = (seed2, s)$
$c =$ HQC.Encrypt(pk, e, θ) [PKE.Ecrypt]
$K := H_1(m, c)$ [Compute K]
$d := H_2(m)$ [Compute d]
return $(K,\ C = (c, d, salt))$

HQC KEM Decapsulation($sk, \bar{C}, d, salt$)

$m' =$ HQC.Decrypt(sk, C) [PKE.Decrypt]
Compute $\theta' \leftarrow G(m', seed2, salt)$ [Re-encryption and compute K]
Compute $c' =$ HQC.Encrypt(pk, m', θ')
Check if $c = c'$ or $d = H_2(m')$, if false abort (return $\perp$)
Compute $K = H_1(m', c')$
return (K)

The QFO$^{\perp}$ transformation applied in HQC is differentiated from that applied to BIKE and Classic McElice in that, as it is a transformation with explicit rejection, no pseudo-random sequence is added to the secret key and, we have commented, the key generation algorithm remains identical for PKE and KEM. In the encapsulation, we have, as in the previous cases, that the shared secret is defined as $K := K(m, c)$. The major difference is in decapsulation, where after re-encrypting the message, in the case that the generated cipher is not equal to the received cipher, $\perp$ is the output of the algorithm. Furthermore, it can be seen in the Algorithm 3 during the encapsulation is defined $d := H_2(m)$ and it used in the decapsulation as a signature of e to check its veracity.

4 Performance analysis

After the security of the algorithms, the second most important characteristic to evaluate is their performance. Performance is measured, principally, in two ways: (1) the length of the ciphertexts and keys (in bytes) and (2) the time of execution of the algorithms (in cycles). Clearly, the most desirable situation is to have highly secure algorithms and with good performance. Code-based algorithms like McEliece were known for a long time, before they were considered as candidates in the NIST call. The main reason behind the lack of attention to these algorithms in past years comes from their high computational cost and the large length of their keys.

The length of the ciphertexts and keys is a fixed amount for each parameter set of each algorithm and such information was presented in each submission. In Table 3 we gather all the data, based in the target security of each parameter set. In the case of Classic McEliece there are

TABLE 3. Public and secret keys, and ciphertext size comparison (in bytes) for code-based algorithms

Data	Sec. Level	McEliece	BIKE	HQC
	1	261 120	1 540	2 249
pk	3	524 160	3 082	4 522
	5	1 044 992	5 122	7 245
	1	6 492	280	40
sk	3	13 608	418	40
	5	13 932	580	40
	1	96	1 572	4 497
C	3	156	3 114	9 042
	5	208	5 154	14 485

six parameter sets targeting the level 5 security, but we only present the sizes for the parameter set kem/mceliece6688128, which is the one that we have considered in our performance analysis.

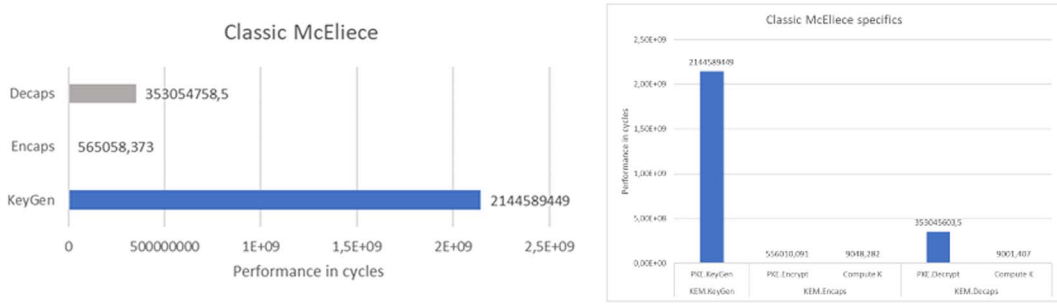
In the submission of each algorithm the times of execution of the KEM algorithms (key generation, encapsulation and decapsulation) are also included. However, since the data were obtained using different devices and architectures, it is not reasonable to establish comparisons between them. In fact, the computers used to run BIKE and HQC were considerably different, and the one used in BIKE had very limited computing power. To be able to make a fair comparison of the algorithms, we will run all three on the same device and architecture.

The platform used for all the tests was Ubuntu 20.04.1 LTS, Intel Core i7 – 4790 CPU @3.60Ghz; 4 Cores; 8 Threads, RAM 16Gb, motherboard Q87M-E (ASUSTeK). To obtain consistent measurements of the cycles, the Hyper-Threading, Tubo Boost and SpeedStep features were disabled in the platform. We run a test measuring the number of cycles that takes to execute each process presented in the Algorithms 1–3 of Section 3. The test runs a total of 1000 times, but to avoid bias from the background task running on the platform, each instance has been repeated 100 times and has been averaged. We used the NIST reference implementation of the algorithms (https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions) and all the tests we present are for the parameter sets targeting the security level 5.

We have noted that in the encapsulation algorithm of HQC, the primitives involved in defining the pseudorandomly chosen message were static, generating always the same shared secret. To avoid this behaviour, we used the C++ random generator as a good enough replacement for this analysis, only for computational reasons. Next, we present our results of the performance analysis of the three code-based proposals considered in this work. To concretize the results, we show in several figures the computational effort needed for each proposal.

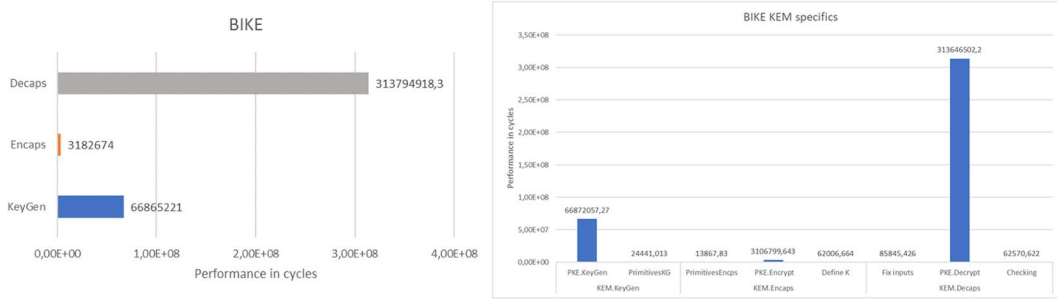
In Figure 1a we show the number of cycles needed to execute each algorithm for the McEliece KEM (see Algorithm 1). It can be appreciated that the larger value is for the KeyGen, whereas the Encaps is the shorter one with a big difference. Moreover, in Figure 1b, the number of cycles for each called function for each algorithm, is showed. For the Encaps algorithm such functions are PKE.Encrypt and Compute K; whereas for the Decaps algorithm they are PKE.Decrypt and Compute K. As expected, the largest computation time is for the KeyGen algorithm.

In Figure 2a one can observe that the number of cycles to compute the Decaps algorithm is much bigger than for the other two. In Figure 2b the number of cycles of the different functions



(a) KEM algorithms (b) Functions in Algorithm 1

FIGURE 1. Classic McEliece KEM performance



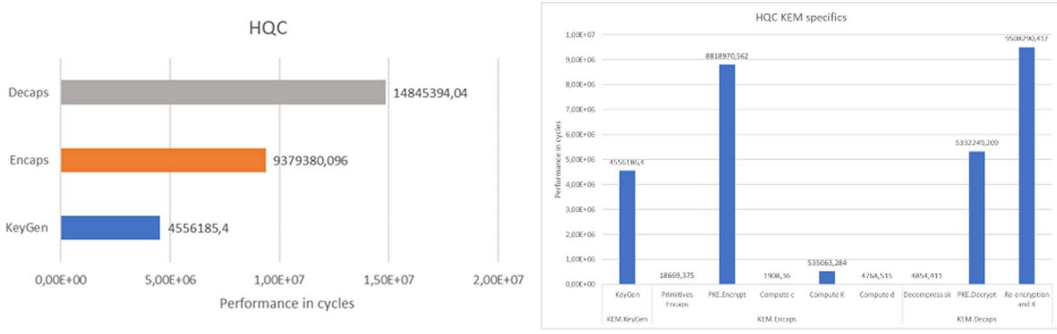
(a) KEM algorithms (b) Functions in Algorithm 2

FIGURE 2. BIKE KEM performance

called for each algorithm is showed. For the KeyGen algorithm (see Algorithm 2) the functions are PKE.KeyGen and PrimitivesKG; for the Encaps algorithm the functions are PrimitivesEncps, PKE.Encrypt, and Define K; whereas for the Decaps algorithm they are Fix inputs, PKE.Decrypt and Checking. Clearly, the PKE.Decrypt is the function that needs the largest computation time.

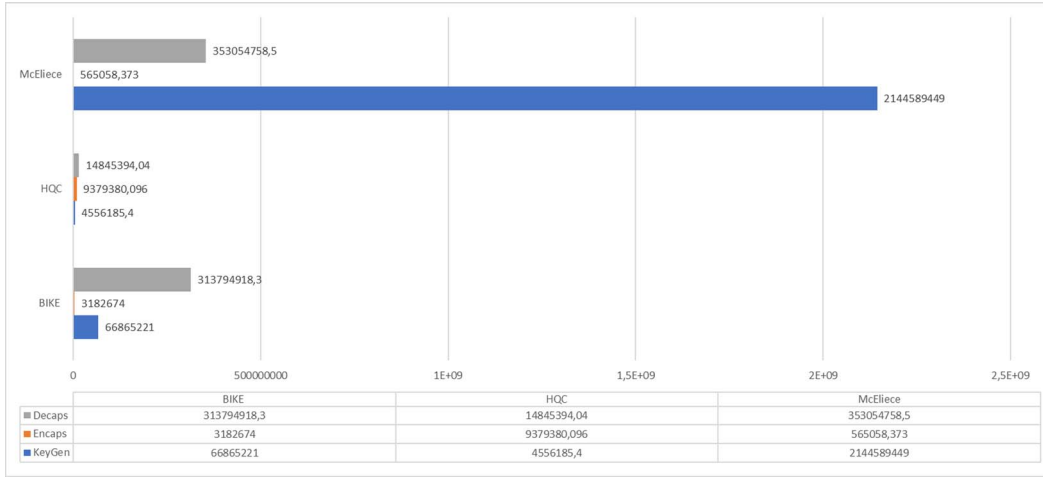
In Figure 3a we show similar results for the HQC proposal (Algorithm 3). In this case, the three algorithms show a step increase, being the Decaps the one which takes the longest time to be executed. On the other hand, in Figure 3b, we show the results obtained for the functions in the HQC proposal. Here, the Encaps algorithm calls the functions Primitive Encaps, PKE.Encrypt, Comnpute_c, Compute_K and Compute_d, whereas Decaps calls decompress, PKE.Encrypt, and Re-encryption and K. The process that takes the longest time is the Decaps algorithm, while the KeyGen is the less time-consuming one.

Finally, Figure 4 shows the execution time, in cycles, of the KEM algorithms for Classic McEliece, BIKE, and HQC proposals. From it, it can be deduced that the worst case corresponds to Classic McEliece because it needs more computation time than the other two proposals. Moreover, it is worthwhile to mention the very large time that it uses to generate its keys, which is due to the large size of them. In any case, the best behavior in relation to the computational effort corresponds to HQC.



(a) KEM algorithms (b) Functions in Algorithm 3

FIGURE 3. HQC KEM performance



	BIKE	HQC	McEliece
Decaps	313794918,3	14845394,04	353054758,5
Encaps	3182674	9379380,096	565058,373
KeyGen	66865221	4556185,4	2144589449

FIGURE 4. Performance comparison (cycles) between McEliece, BIKE and HQC

5 Conclusions

We have studied the code-based proposals of the NIST call, Classic McEliece, BIKE and HQC—in particular, how they apply the FO transformation and their performance. From a security standpoint this is a crucial analysis, since all three algorithms studied rely in that transformation to obtain, not only IND-CCA security, but also security in a QROM. In our previous study [7], we mentioned the possibility of changes in the transformation applied to the code-based algorithms in future versions. In fact, Classic McEliece has changed this. It is still remarkable that some of the security reductions quoted in the algorithms submissions do not correspond with the exact version of the transformation applied.

While the FO is critical for security, it only affects significantly the performance of the algorithms when the re-encryption is required. The case of HQC is a good example, in Figure 3 it can be appreciated how the decapsulation takes extra time just to check if the result of the decryption

is correct. Also, the re-encryption is a target for side-channel attacks, hence the versions of the transformation without re-encryption are preferable.

Our performance analysis allows to compare the three candidates studied. Clearly, HQC presents the best results among the three algorithms, since it has the lowest execution time and the public key sizes are of the same order than those of BIKE (much smaller than in the case of Classic McEliece). Classic McEliece is the algorithm with the longest computational times and also longest keys. However, it is a well-known and studied algorithm, without clear weakness known so far. This makes McEliece more reliable than the other algorithms.

Acknowledgements

This work was supported in part by project ORACLE, PCI2020-120691-2, funded by MCIN/AEI/10.13039/501100011033, in part by QURSA project TED2021-130369B-C33 funded by MCIN/AEI/10.13039/501100011033, both of them co-funded by the European Union 'NextGenerationEU'/PRTR, in part by P2QProMeTe (PID2020-112586RB-I00), funded by MCIN/AEI/10.13039/501100011033, and in part by the EU Horizon 2020 research and innovation programme, project SPIRS (grant agreement no. 952622).

References

[1] C. Aguilar, N. Melchor, S. Aragon, L. Bettaieb, O. Bidoux, J.C. Blazy, P. Deneuville, E. Gaborit, G. Persichetti, J. Zémor. HQC (Hamming Quasi-Cyclic). Online publication (2021), https://pqc-hqc.org/

[2] M.R. Albrecht, D.J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V.M. an Ingo von Maurich, R. Misoczki, R. Niederhagen, K.G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C.J. Tjhai, M. Tomlinson and W Wang. Classic McEliece: conservative code-based cryptography (2020), https://classic.mceliece.org/nist.html

[3] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. Aguilar Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.P. Tillich, V. Vasseur and G. Zemor. BIKE (Bit Flipping Key Encapsulation). Online publication (2021), https://bikesuite.org

[4] D. J. Bernstein and E. Persichetti. Towards KEM unification. *Cryptology ePrint Archive*, Report 2018-526, 2018. https://eprint.iacr.org/2018/526.

[5] N. Drucker and S. Gueron, et al. On the applicability of the Fujisaki-Okamoto transformation to the BIKE KEM. *Cryptology ePrint Archive*, Report 2020-510, 2020. https://eprint.iacr.org/2020/510.

[6] E. Fujisaki and T. Okamoto.: Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the 19th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO'99, Lecture Notes Comput. Sci., vol. 1666*, pp. 537–554 (1999), https://doi.org/10.1007/3-540-48405-1_34

[7] M. González, L. de la Torre and S. Hernández Encinas. About the Fujisaki-Okamoto transformation in the code-based algorithms of the NIST post-quantum call. In *Proceedings of the International Joint Conference, 15th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2022), 13th International Conference on European Transnational Education (ICEUTE 2022), Lecture Notes Netw. Syst. vol. 523*, pp. 75–85, 2022.

[8] M. González de la Torre, L. Hernández Encinas and A. Queiruga-Dios. Analysis of the FO

transformation in the lattice-based post-quantum algorithms. *Mathematics*, **10**, 2967, 2022. https://doi.org/10.3390/math10162967.

[9] D. Hofheinz, K. Hövelmanns and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Proceedings of the 15th International Conference Theory of Cryptography TCC'2017, Lecture Notes Comput. Sci., vol. 10677*, pp. 341–371, 2017.

[10] H. Jiang, Z. Zhang, L. Chen, H. Wang and Z. Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. *Cryptology ePrint Archive*, Report 2017-1096, 2017. https://eprint.iacr.org/2017/1096.

[11] R. McEliece. A public key cryptosystem based on algebraic coding theory. *Technical Report*, 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

[12] NIST Post-quantum cryptography. On-line publication (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

[13] T. Saito, K. Xagawa and T. Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2000, Lecture Notes Comput. Sci., vol. 10822*, pp. 520–551, 2018.

[14] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, **41**, 303–332, 1999. https://doi.org/10.1137/S0036144598347011.

Received 28 March 2024