DOI: 10.2478/ias-2025-0004 • JIAS • 20(2) • 2025 • 58-73

# A Novel Approach to Bit-Flipping Threshold Selection with Particle Swarm Optimization in the QC-MDPC-McEliece Cryptosystem

**Abdellatif Kichna[1*], Abderrazak Farchane and Said Hakimi**

[1] LIMATI Laboratory, Polydisciplinary Faculty, Sultan Moulay Slimane University, P.O. Box 592, Beni Mellal 23000, Morocco

*Correspondence author: abdellatif.kichna@usms.ac.ma

**Abstract:** The advent of quantum computers brings about the need for cryptosystems that can withstand quantum attacks. The QC-MDPC based McEliece cryptosystem is one such post-quantum cryptographic scheme, offering robust security yet posing significant challenges in efficient decoding. Central to these challenges is the selection of an optimal threshold for bit-flipping decoding algorithms. This paper presents a novel approach that applies Particle Swarm Optimization (PSO) to dynamically establish optimal thresholds, thereby aiming to minimize the number of iterations needed for successful decoding. We demonstrate the effectiveness of our method through rigorous simulations, underscoring its potential to enhance the efficiency of the McEliece cryptosystem and comparable post-quantum cryptographic schemes. This research could mark a substantial step towards greater practicality in the realm of post-quantum cryptography.

**Keywords:** bit flipping decoding, Mceliece cryptosystem, particle swarm optimization, post-quantum cryptography, QC-MDPC codes

## 1 Introduction

The constant evolution and growing sophistication of computing power pose an increasing threat to the security of information in the digital age [1]. Cryptographic systems are at the forefront of protecting sensitive information, making their robustness and resilience against computational attacks critically important. One significant development in the field of cryptography is the advent of quantum computers, which could potentially render many of today's cryptographic systems vulnerable [2]. In response to this, post-quantum cryptography has emerged as a discipline focusing on cryptographic algorithms believed to be secure against an attack by a quantum computer [3].

The McEliece cryptosystem is one such post-quantum cryptographic scheme [4]. It has stood the test of time since its invention in the late 20th century, demonstrating resilience against known quantum attacks. A key component of this cryptosystem is the decoding of specific error-correcting codes known as QC-MDPC codes [5]. While these codes provide the robust security necessary for the McEliece cryptosystem, they also present a significant challenge in terms of efficient decoding. Recent work has deepened our understanding of QC-MDPC decoding challenges. Sendrier and Vasseur [6] rigorously analyzed the decoding failure rate of bit-flipping decoders, identifying critical dependencies on code parameters and error distributions. Their findings underscore the need for adaptive strategies to maintain robustness across diverse noise conditions. Vasseur [7] further dissected the interplay between parity-check equations and iterative decoding, providing theoretical bounds on error correction. Separately, Nosouhi et al. [8] demonstrated the practicality of bit-flipping decoding in post-quantum key encapsulation, highlighting their efficiency-security trade-offs. However, the critical parameter of dynamic threshold selection has remained underexplored and largely heuristic. The role of a decoding threshold in these algorithms is critical, and a sub-optimal choice can have a detrimental impact on decoding efficiency and correctness [9]. Existing bit-flipping threshold strategies, while effective, have limitations. The heuristic rule introduced by Misoczki et al. relies on subtracting a fixed integer from the maximum number of unsatisfied parity-check equations, which may not be universally applicable or optimal for different system configurations or noise levels. This deterministic approach can lead to decoding failures in certain scenarios and may not adapt well to varying error patterns.

The focus of this research is to address this challenge using an innovative approach. We introduce a decoding strategy for QC-MDPC codes that leverages the power of Particle Swarm Optimization (PSO), a strategy for optimization that draws inspiration from natural processes [10]. PSO's unique capacity to navigate a broad search space and hone in on optimal solutions [11, 12] is utilized here in a novel context - to determine the optimal thresholds in bit-flipping decoding algorithms. By reducing the iteration count required for successful decoding, this research could contribute to enhancing the efficiency of the McEliece cryptosystem.

Particle Swarm Optimization (PSO) emerges as a promising solution due to its ability to dynamically adjust threshold values based on the evolving decoding conditions. Unlike deterministic methods, PSO is a stochastic optimization technique that can explore the solution space more thoroughly and adaptively. Its strength lies in balancing exploration and exploitation, making it suitable for complex, non-linear optimization problems like threshold selection in bit-flipping decoding. PSO's fast convergence rate and gradient-free nature make it particularly appropriate for this application, as the decoding process requires rapid identification of optimal solutions without prior knowledge of the error distribution. This work introduces a paradigm shift by framing threshold selection as an optimization problem, leveraging PSO to dynamically determine thresholds tailored to each decoding iteration. To our knowledge, this is the first application of swarm intelligence to threshold optimization in cryptographic decoding, addressing a critical gap in the QC-MDPC literature.

In the following sections, we delve deeper into the intricacies of QC-MDPC code decoding, discuss the concept and workings of the PSO technique, and present our novel decoding methodology. We then provide a detailed account of our simulation setup and discuss the obtained results, followed by a comparison with existing methods. We conclude with our reflections on the potential impact of this research and a perspective on potential avenues for subsequent research.

## 2 Background and Related Work

### *2.1 QC-MDPC-McEliece Cryptosystem*

The QC-MDPC based McEliece cryptosystem is a variant of the classic McEliece cryptosystem that uses quasi-cyclic moderate density parity check (QC-MDPC) codes [5]. It was proposed to address the significant challenge of large key sizes in the original McEliece cryptosystem, without compromising the system's notable resistance to quantum attacks. The QC-MDPC variant uses efficient key generation and encryption/decryption procedures, making it a strong contender in the field of postquantum cryptography.

#### 2.1.1. *Generation of Keys*

We start the keys generation process for a QC-MDPC McEliece system, aspiring for a predetermined security level, symbolized as λ. Key parameters, namely $n_0$, n, r, and ω, are selected corresponding to the desired security level λ.

Initially, we construct a random QC-MDPC code. This is accomplished by randomly selecting a vector $h$ in $\mathbb{F}_2^n$ domain, which maintains a weight of ω. We partition $h$ into $n_0 = \frac{n}{r}$ equal segments, represented as $h = \left[ h_0 \middle| h_1 \middle| \cdots \middle| h_{n_0-1} \right]$. The parity check matrix H is constructed subsequently as follows:

$$\mathrm{H} = \left[ H_0 \middle| H_1 \middle| \cdots \middle| H_{n_0-1} \right], \tag{1}$$

Here, each $H_i$ constitutes a cyclic matrix, with $h_i$ as the first row. A cyclic matrix is a square matrix where each row (or column) is a cyclic permutation of the previous row (or column). This means that each subsequent row (or column) is obtained by shifting the elements of the previous row (or column) cyclically to the right (or left). To ensure the accuracy of the process, the block $H_{n_0-1}$ must be invertible. If this condition is not met, the procedure is restarted with a new randomly chosen h. H is kept as the private key. The public key, denoted as G, is derived from H as:

$$G = \left( I \middle| Q \right), \tag{2}$$

where I is the identity matrix and Q is:

$$Q = \begin{pmatrix} \left(H_{n_0-1}^{-1} H_0\right) \\ \left(H_{n_0-1}^{-1} H_1\right) \\ \vdots \\ \left(H_{n_0-1}^{-1} H_{n_0-2}\right) \end{pmatrix} \tag{3}$$

### 2.1.2. *Encryption process*

Encryption is a fundamental step in the QC-MDPC McEliece Cryptosystem that allows secure communication of information. The process begins with a message vector m that resides in the finite field $\mathbb{F}_q^k$. This vector holds the actual information to be securely transmitted.

To obfuscate this information and protect it from potential eavesdroppers, a unique random error vector e is generated, residing in the finite field $\mathbb{F}_q^n$. This error vector serves a crucial purpose in introducing 'noise' into the system. Importantly, this vector is characterized by a weight ≤ t, where t denotes the maximum number of errors that the QC-MDPC code can handle and correct.

The error vector e and message vector m are then used in conjunction to generate the ciphertext - the scrambled, unintelligible version of the message that is safe to transmit over the public channel. This process is carried out using the public key matrix G, a significant component of the McEliece Cryptosystem. The relationship is expressed succinctly by the equation c = mG+e, where the ciphertext c is derived from the multiplication of the message vector m and the public key matrix G, with the error vector e added, as illustrated in Figure 1.

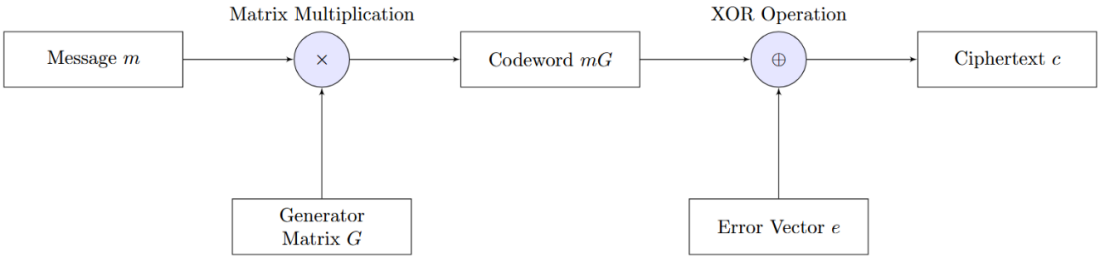


***Figure 1.*** Encryption process. Plaintext m is encoded into codeword mG, combined with an error vector e to produce ciphertext c=mG+e.

The resulting ciphertext vector c now effectively disguises the original message m, making it secure for transmission. It's worth emphasizing that this encryption process leverages the mathematical properties of finite fields and the structure of the QC-MDPC McEliece Cryptosystem to ensure strong cryptographic security.

### 2.1.3. *Decryption process*

The decryption process, also known as decoding, is the mirror image of the encryption process - it is here that the original message m is recovered from the transmitted ciphertext $c \in \mathbb{F}_2^n$. This delicate procedure of reversing the encryption process is crucial for any cryptosystem, and the QC-MDPC McEliece Cryptosystem is no exception. After error correction via the bit-flipping algorithm, the original message m is extracted directly from the first r bits of the corrected codeword. This is enabled by the systematic form of the public key matrix G (eq. 2). In systematic encoding, the message m is embedded unchanged in the first r positions of the codeword c = mG+e. Once errors in e are corrected, m can be trivially recovered by truncating the codeword to its first r bits (Figure 2).

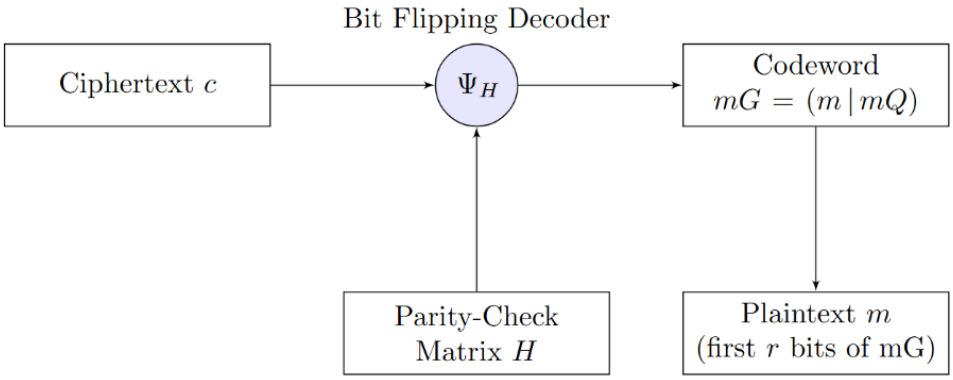


***Figure 2.*** Decryption process. Ciphertext c is decoded via bit-flipping decoder. Corrected codeword mG yields m from its first r bits, exploiting G's systematic structure.

To correctly unravel the ciphertext and reveal the original message, a decoding algorithm is essential. A prominent method suggested in [5] is the bit flipping algorithm, which is a technique originally proposed by Robert Gallager in 1963 for decoding Low-Density Parity Check (LDPC) codes, which are a type of error correcting codes [13].

At the heart of this algorithm lies a basic but powerful observation: each bit in the syndrome, a mathematical expression that captures the relationship between the received and transmitted codewords, reveals whether its corresponding equation is satisfied or not. If an equation is not satisfied, it signifies an error in the transmitted code [14].

In practice, positions in the received code that are implicated in a high number of unsatisfied equations are likely to be errors. Recognizing this, the bit-flipping algorithm iteratively flips these suspect bits based on a predefined threshold. The threshold is a predetermined value used to determine whether a bit in the received vector should be flipped during the decoding process. The algorithm compares the magnitude of the error associated with each bit to this threshold. If the error magnitude exceeds the threshold, the algorithm flips the corresponding bit in an effort to reduce the overall number of unsatisfied equations. This iterative process aims to home in on the transmitted code and, ultimately, the original message.

### 2.1.4. *The role of the Threshold in the Bit Flipping decoder*

In [5], a heuristic rule was introduced to define the threshold, primarily based on the maximum number of unsatisfied parity-check equations (UPC) at a given iteration, denoted as max(UPC). A constant $\delta$ value is subtracted from max(UPC) to set the bit-flipping threshold. The choice of $\delta$ is a crucial aspect of this method, with the suggested value of 5 obtained empirically for their specific system configuration and conditions as determined by Misoczki et al. For example, in our experiments with a security level of 80, the initial maximum number of unsatisfied parity-check equations (max(UPC)) ranged from 29 to 36. This resulted in a threshold of 24 to 31. As decoding progresses and max(UPC) decreases, this threshold also reduces.

The minimum threshold value of 24 occurs when the error pattern leads to the lowest observed max(UPC) of 29. This situation typically arises when errors are distributed in a way that minimizes the number of parity-check equations violated initially. Even with this lower threshold, the algorithm can still identify and correct the most likely error positions while minimizing the risk of flipping correct bits. The reasoning behind this minimum value is to ensure that the decoder remains conservative enough to avoid introducing new errors while still making progress toward correction.

The maximum threshold value of 31 occurs when the error pattern causes the highest observed max(UPC) of 36. This represents more challenging error configurations where more parity-check equations are violated initially. The higher threshold in this case helps prevent excessive bit flipping that could lead to error propagation, while still allowing the algorithm to make progress toward correcting errors. The reasoning behind this maximum value is to provide sufficient flexibility for the decoder to handle dense error patterns without becoming overly aggressive and potentially destabilizing the decoding process.

This represents more challenging error configurations where more parity-check equations are violated initially. The higher threshold in this case helps prevent excessive bit flipping that could lead to error propagation, while still allowing the algorithm to make progress toward correcting errors.

The reasoning behind this maximum value is to provide sufficient flexibility for the decoder to handle dense error patterns without becoming overly aggressive and potentially destabilizing the decoding process.

In effect, this method introduces an adaptive, dynamic thresholding strategy that adjusts the threshold depending on the present condition of the decoding process [15]. At the start of the decoding, when the number of errors and unsatisfied equations is likely to be high, the threshold will also be relatively high. This prevents the algorithm from flipping bits too aggressively, reducing the risk of making premature or incorrect corrections.

As the decoding process progresses and the number of unsatisfied equations decreases, max(UPC) and thus the threshold will also decrease. This allows the algorithm to become more aggressive in flipping bits, helping to clear up residual errors.

The advantage of this method is its adaptability to varying decoding conditions. By making the threshold proportional to max(UPC), the decoder's behavior is made flexible. When there are many errors, it is cautious; when there are few errors, it is assertive. This makes the decoder robust to a variety of error conditions and helps maintain reliable performance.

However, despite its efficiency, the bit-flipping decoding method is not without its drawbacks. One potential issue with this approach lies in its deterministic nature. In certain instances, the algorithm might reach a state of deadlock, where no bit satisfies the flipping condition, yet the syndrome is not zero. This situation represents a decoding failure.

Another concern involves the choice of the $\delta$ value. The proposed $\delta$ of 5, however, is largely heuristic and empirical. Although this value may work well for the specific system configuration and conditions that Misoczki et al. considered, it might not be universally applicable or optimal for other conditions or variations in the system. In other words, different QC-MDPC codes, operating conditions, or noise levels might necessitate different $\delta$ values for optimal decoding performance.

Given these inherent limitations of the heuristic thresholding strategy in the bit-flipping decoding process, there is a significant need to explore more flexible and robust solutions. One such promising approach is the implementation of optimization algorithms that can dynamically adjust the threshold according to the evolving decoding conditions. In this context, Particle Swarm Optimization (PSO) emerges as an attractive proposition.

## 3 Particle Swarm Optimization: An Overview

### *3.1 PSO basics*

Particle Swarm Optimization (PSO) is a heuristic global optimization method introduced by Eberhart and Kennedy in 1995 [10], inspired by the collective dynamics observed in bird migration or fish schooling. The PSO algorithm operates based on the cooperation and intelligence inherent in swarm behavior.

Within the PSO framework, individual solutions are metaphorically visualized as 'birds' navigating the search field, known as'particles.' Each particle, denoted by $i$, explores the solution space within $d$ dimensions. The position of particle $i$ is represented as $x_{id}$, and its velocity as $v_{id}$. Notations such as the inertia weight ($\omega$), cognitive parameter ($c_1$), social parameter ($c_2$), and random numbers ($r_1$ and $r_2$) ranging from 0 to 1 are integral to the algorithm. Personal best ($p_{id}$) and global best ($p_{gd}$) positions, as well as the fitness function ($f(x_{id})$) at the current position of a particle, further contribute to the PSO dynamics. Additionally, the termination condition, often defined in terms of a specific number of iterations or a fitness threshold, shapes the orchestration of the cooperative exploration of the solution space by the swarm of particles.

### *3.2 Basic algorithm steps*

The basic PSO algorithm includes the following steps (see Figure 3):

- Initialization: Commence with a cluster of particles, each endowed with random placements and velocities across $d$ dimensions within the problem scope. The location of each particle is characterized by a vector, symbolizing a probable solution. Likewise, each particle's velocity is signified by a vector, indicating the trajectory of the particle's motion. Moreover, each particle is equipped with a memory function, preserving its optimal achieved position, along with the globally optimal position attained by any particle.

- Fitness Evaluation: For every particle, compute the fitness value tied to its present location. If this position surpasses its past optimum, the best position gets updated. Analogously, should the present location outperform the global best, the global best position is refreshed.
- Velocity and Position Update: Revise the velocity and location of every particle according to the subsequent formulas:

$$V_i(t+1) = \omega . V_i(t) + c_1 . r_1 . \left(P_i(t) - X_i(t)\right) + c_2 . r_2 . \left(P_g(t) - X_i(t)\right)_{id}, \tag{4}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{5}$$

Where $V_i(t)$ represents the particle's velocity, $\omega$ stands for the inertia weight, $c_1$ and $c_2$ denote cognitive and social parameters, $r_1$ and $r_2$ are random numbers ranging from 0 to 1, $p_i(t)$ and $P_g(t)$ correspond to the personal best and global best positions, respectively, and $X_i(t)$ indicates the particle's current location.

- Termination: Continue executing steps 2 and 3 until a predetermined stopping condition is achieved. This could be the completion of a specific number of iterations or the attainment of a fitness value that meets or exceeds a particular threshold.

**Algorithm 1** Particle Swarm Optimization (PSO)

**Initialization:**
Initialize swarm of $N$ particles, each with random positions $x_{id}$ and velocities $v_{id}$ in $d$-dimensional space.
Set personal best positions $p_{id} = x_{id}$ for all particles.
Determine global best position $p_{gd}$ from $p_{id}$ with the best fitness.
**for** each iteration until termination condition is met **do**
    **for** each particle $i$ **do**
        Compute fitness function $f(x_{id})$.
        **if** $f(x_{id}) < f(p_{id})$ **then**
            Update personal best: $p_{id} = x_{id}$.
        **end if**
        **if** $f(p_{id}) < f(p_{gd})$ **then**
            Update global best: $p_{gd} = p_{id}$.
        **end if**
        Update velocity:

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})$$

        Update position:

$$x_{id} = x_{id} + v_{id}$$

    **end for**
**end for**
**Return** best global solution $p_{gd}$.

***Figure 3.*** PSO pseudocode.

### *3.3 Advantages of PSO*

PSO has a fast convergence rate and it does not require the gradient of the problem, making it suitable for non-differentiable optimization problems [16]. Furthermore, PSO has been applied in various fields, such as neural network training, fuzzy system control, and other areas of engineering.

## 4 PSO-Based Bit-Flipping Threshold Selection Algorithm

This section delineates the methodology used to apply Particle Swarm Optimization (PSO) in the domain of QCMDPC code decoding (see Figure 4). Our proposed methodology focuses on leveraging the heuristic capabilities of PSO to determine optimal thresholds dynamically during the iterative bit flipping process.

Given the non-trivial nature of the threshold selection problem for bit flipping decoding, a method is needed that can not only find optimal threshold values but also adapt to changes in the solution space. This is where the concept of PSO comes into play, with its inherent capability to search across a broad search space and adapt dynamically.

To adapt PSO to the threshold selection problem in bit flipping decoding, we employ the particles to symbolize potential threshold values. In this representation, each particle stands for a unique threshold value at a given bit-flipping iteration. As the swarm of particles moves iteratively through the decoding process, the position of a particle represents a particular threshold value, while its velocity - indicating the particle's movement in the 'threshold value space' - is updated as the search advances.

### *4.1 Particle representation and update*

In decoding QC-MDPC codes, the choice of threshold in the bit-flipping algorithm is crucial. A poorly chosen threshold may cause incorrect bit flips or decoding failure. We propose a PSO-based approach to dynamically select an optimal threshold during each iteration of the decoding process.

Each particle in our swarm represents a candidate threshold value. The swarm evolves to minimize the syndrome weight — a measure of how close the current codeword is to a valid code. The update rules are crafted so as to enable the swarm to traverse the search space effectively, and are defined as follows:

- Velocity update rule:

$$V_i(t+1) = \omega.\, V_i(t) + c_1.\, r_1.\left(P_i(t) - X_i(t)\right) + c_2.\, r_2.\left(P_g(t) - X_i(t)\right), \tag{6}$$

Here, $V_i(t)$ denotes the velocity of particle $i$ at time $t$, $\omega$ is the inertia weight, $c_1$ and $c_2$ are the cognitive and social learning factors, $r_1$ and $r_2$ are random numbers, $P_i(t)$ is the best threshold value achieved by particle $i$ at time $t$, $X_i(t)$ is the current threshold value of particle $i$ at time $t$, and $P_g(t)$ is the best threshold value discovered by the swarm at time $t$.

This rule incorporates three components that guide the search for optimal thresholds. The inertia weight ($\omega$) maintains a balance between global and local exploration, allowing the algorithm to initially explore a broad range of threshold values and gradually focus on promising regions as the decoding process progresses. The cognitive component $c_1.\, r_1.\, (P_i(t) - X_i(t))$ enables each particle to learn from its own experience, refining its threshold value based on previously successful solutions. The social component $c_2.\, r_2.\, (P_g(t) - X_i(t))$ facilitates knowledge sharing among particles, steering the swarm toward globally optimal threshold values that have proven effective in correcting errors across different iterations.

- Position update rule:

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{7}$$

Directly adjusts the threshold value for the next iteration based on the updated velocity. This adaptive mechanism allows the PSO algorithm to dynamically determine threshold values that are optimal for each specific decoding scenario, taking into account the current state of the syndrome and the distribution of unsatisfied parity-check equations. By iteratively applying these update rules, the PSO algorithm can efficiently navigate the solution space, avoiding local optima and converging on threshold values that minimize the number of iterations required for successful decoding.

These rules provide an intelligent, guided mechanism for the swarm to explore and exploit the space of threshold values, enabling it to dynamically determine optimal thresholds for each bit-flipping iteration. By integrating these update rules into the iterative decoding process, our approach strives to improve the efficacy of the bit-flipping decoder by reducing the syndrome weight and thus, getting closer to the original transmitted codeword.

### *4.2 Fitness Function: Minimizing Syndrome Weight*

To gauge each particle's performance and drive the direction of the search, we use a fitness function grounded on the number of bit flips in the decoded message. The fitness of a particle is the syndrome weight accomplished by using the particle-represented threshold value. The algorithm's end goal is to

discover a threshold that minimizes the syndrome weight, signalling that the decoded message is as similar as possible to the original message. Consequently, the fitness function and syndrome weight are directly associated, with the former serving as an indicator of the threshold's decoding performance.

Below is an outlined process of using syndrome weight as a fitness function within the PSO algorithm:

1. Initialize a particle swarm: We generate a random array of particles where each one represents a potential threshold value for the corresponding bit-flipping iteration.
2. Evaluate each particle's fitness: We calculate the syndrome weight by adding the actual syndrome with the corresponding columns of the potential bits that have a higher count of unsatisfied parity checks than the current threshold. Here, the fitness of each particle is defined as the syndrome weight, and our aim is to minimize this value to zero.
3. Update particle positions and velocities: Based on the fitness values of the particles, we apply the PSO algorithm to update their positions and velocities.
4. Repeat steps 2-3 until a convergence point is reached: This process is reiterated until we either reach a convergence point, complete a predefined number of iterations, or achieve a syndrome weight of zero. Typically, convergence is noted when there is no significant improvement in the lowest fitness value of the swarm.
5. Select the optimal particle: We select the particle with the lowest fitness value as the solution for that specific bit-flipping iteration. Note that the chosen threshold value is unique to the current decoding operation and is not reused for future noisy codewords due to the unique nature of each ciphertext and corresponding parity check matrix.

By implementing the PSO-based methodology presented above, it is possible to optimize the threshold values in the bit flipping decoding process, thereby enhancing the efficiency and performance of decoding QC-MDPC codes.

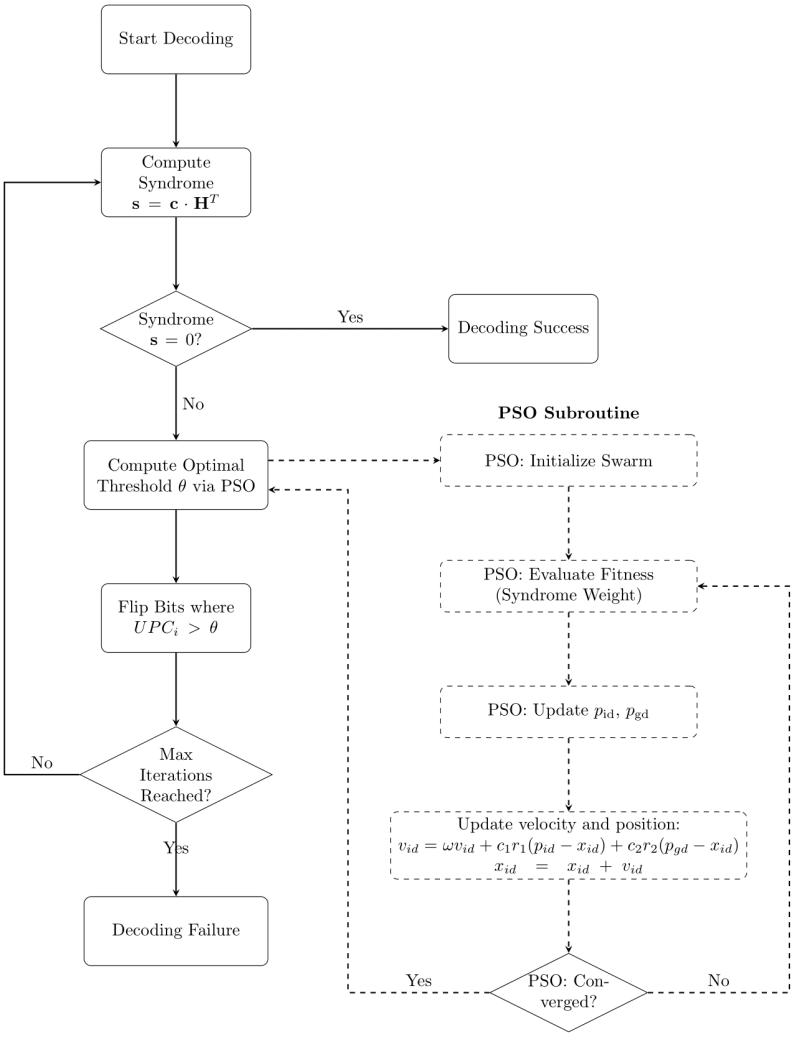


***Figure 4.*** PSO based threshold selection flowchart.

### *4.3 Formal proof statement*

The threshold selection problem is formally defined as a combinatorial optimization task to minimize syndrome weight $S_w$ across decoding iterations:

$$\tau^* = arg \min_{\tau} s_w(\tau),\ \tau \in [\tau_{min},\ \tau_{max}], \tag{8}$$

Where $\tau_{min}$ and $\tau_{max}$ are derived from the observed range of unsatisfied parity-check equations (UPC) during decoding. This formulation transforms threshold selection into a bounded, non-linear optimization problem. Traditional gradient-based methods are ill-suited here due to the discrete, non-differentiable nature of syndrome weight minimization. Particle Swarm Optimization (PSO), however, thrives in such spaces by leveraging stochastic exploration and swarm intelligence to navigate potential solutions without relying on gradient information

### *4.4 Theoretical analysis of computational complexity*

A theoretical analysis of the computational complexity of the PSO-based threshold selection method reveals its efficiency compared to traditional approaches. Each iteration of the PSO algorithm involves evaluating the fitness of all particles, updating their velocities and positions, and selecting the optimal threshold. The fitness evaluation for each particle requires calculating the syndrome weight, which has a complexity of $O(n)$, where n is the code length. For m particles and k iterations, the total complexity of the PSO algorithm is $O(m.k.n)$. In contrast, the traditional bit-flipping algorithm has a complexity of $O(n.t)$, where t is the number of iterations required for convergence. While the PSO-based method introduces additional overhead due to the swarm optimization process, its ability to dynamically optimize the threshold often results in fewer iterations $k$ compared to the traditional method $t$. This trade-off between computational overhead and reduced iterations makes the PSO-based approach more efficient in practice, particularly for complex decoding scenarios. Furthermore, the parallel nature of the PSO algorithm allows for potential optimizations in hardware implementations, further enhancing its practical efficiency.

## 5 Simulation and Results

In this section, we present an in-depth overview of the experimental setting utilized to assess the efficacy of our proposed Particle Swarm Optimization (PSO)-based approach for enhancing the bit flipping decoding algorithm in the context of the McEliece cryptosystem based on Quasi-Cyclic Moderate-Density Parity-Check (QC-MDPC) codes.

### *5.1 Experimental setup*

A vital aspect of our study is the experimental phase, intended to critically evaluate and verify the merits of our proposed approach. All experiments were conducted using MATLAB R2016a for numerical computations, visualization, and implementation of the PSO algorithm, performed on a resilient system equipped with an Intel Core i7-7500U CPU operating at 2.70GHz, complemented with 8GB of RAM, thereby ensuring an adequate computational environment for our purpose.

Primarily focused on achieving a security level of 80, in line with recommended parameters depicted in Table 1. The key sizes are determined by the code parameters (n, k, t) which are selected to achieve a specific security level against the best-known attacks. The code length n = 9602 and dimension k = 4801 are chosen to balance security and efficiency. The error correction capability t = 84 is selected to ensure the system can correct the maximum number of errors while maintaining practical decoding performance.

These parameters result in key sizes that represent a significant improvement over the original McEliece cryptosystem while still providing post-quantum security guarantees. The quasi-cyclic structure of the codes helps reduce the key sizes compared to the original McEliece proposal, making the system more practical for real-world applications. The specific values are necessary to maintain the security level against both classical and quantum attacks, with the public and private keys being sufficiently large to prevent brute-force attacks while remaining manageable for implementation.

***Table 1.*** Recommended parameters for QC-MDPC McEliece cryptosystem by Misoczki et al. (2013)

| Security level | $n_0$ | n | r | ω | t |
|---|---|---|---|---|---|
| 80 | 2 | 9602 | 4801 | 90 | 84 |

We generated a total of 10,000 plaintexts and corresponding error patterns, which followed a uniform distribution, ensuring a broad representation of potential real-world scenarios. The plaintexts used in our experiments were vectors of length 4801 bits, corresponding to the message length k in our QC-MDPC code parameters. These vectors were produced using a cryptographically secure pseudorandom number generator (CSPRNG) to ensure uniformity and avoid biases from structured data. The error patterns were generated by randomly selecting t=84 positions (based on our security parameters) in the ciphertext vector of length n=9602 and flipping the bits at those positions. The t error positions were uniformly randomized across the ciphertext vector of length n=9602, with no spatial clustering or predefined patterns. The experiments were repeated twice to guarantee the consistency of the results and also to identify potential limitations or weaknesses of our methodology.

In our evaluation procedure for each of the 10,000 plaintexts, we followed these steps:

- Encryption: Each plaintext was encrypted using the generated public key, a product of our QC-MDPC based McEliece cryptosystem, and then deliberately injected with a random error pattern to enhance security and resist potential attacks.
- Decryption: We then attempted decryption of the artificially corrupted ciphertexts, employing the bit flipping algorithm facilitated by two distinct threshold selection strategies. The first method follows Misoczki et al.'s approach, which deducts an integer $\delta$ from the maximum number of unsatisfied parity check equations. Alternatively, we introduced our novel PSO based threshold method, designed to optimize the threshold value for each iteration of the bit flipping process. This approach leverages the Particle Swarm Optimization (PSO) algorithm, the specific parameters of which are detailed in Table 2. The choice of PSO parameters in our experimental setup was guided by a balance between exploration and exploitation capabilities, as well as computational efficiency. The inertia weight ($\omega = 0.4$) was selected to ensure sufficient global exploration while allowing the swarm to converge relatively quickly on promising threshold values. A lower inertia weight would risk premature convergence, while a higher value might prolong the decoding process unnecessarily. The cognitive constant ($c_1 = 2$) and social constant ($c_2 = 2$) were chosen to equally weight the influence of individual particle experience and collective swarm knowledge. This balance helps maintain diversity in the swarm while still directing the search toward globally optimal solutions. To validate the robustness of our parameter choices, we conducted a sensitivity analysis by varying these parameters within reasonable ranges. Our results indicated that the selected values provided a good trade-off between convergence speed and solution quality. For instance, increasing the cognitive constant beyond 2 led to more erratic particle movements and slower convergence, while decreasing it below 2 resulted in reduced exploration and higher likelihood of premature convergence. Similarly, adjusting the inertia weight demonstrated that values outside the range of 0.3 to 0.5 significantly impacted the algorithm's performance, either by causing excessive iterations or by failing to adequately explore the solution space. The Swarm size was determined empirically to minimize computational overhead while maintaining diversity. Larger swarms (>20) showed diminishing returns in threshold quality versus runtime.

***Table 2.*** Parameters of the Particle Swarm Optimization used in this study

| Parameter | Value |
|---|---|
| Number of particles (swarm size) | 20 |
| Maximum number of iterations | 50 |
| Cognitive constant ($c_1$) | 2 |
| Social constant ($c_2$) | 2 |
| Inertia weight ($\omega$) | 0.4 |
| Random values: $r_1$, $r_2$ | [0,1] |

- Performance Evaluation: Finally, the effectiveness of the two approaches was evaluated by several key metrics, including the required number of bit flipping iterations for successful decryption, the evolution of average syndrome weight over iterations, the total count of flipped bits, and the tally of corrected bits attributed to each method.

## *5.2 Performance comparison*

This section delves into the experimental results comparing the standard bit flipping algorithm as per Misoczki et al.'s approach and the proposed PSO-optimized bit flipping algorithm. We assess the effectiveness of these two strategies with respect to their success rates, error correction rates, and overall computational efficiency. We also scrutinize specific metrics such as the total number of bits flipped and the number of iterations required for complete error correction, which are crucial in understanding the efficiency of these methods. This critical appraisal provides insights into the potential advantages of employing PSO optimization within the McEliece cryptosystem.

Our experimental findings demonstrate the superiority of the PSO-optimized bit flipping algorithm over the standard approach. As shown in Figure 5, the PSO-based method achieves significantly faster convergence:

- Iteration 1: PSO approach achieves a syndrome weight of approximately 1488.2, while the standard method achieves 1566.4.
- Iteration 2: PSO reduces syndrome weight to 614.8, while the standard method reaches 1146.1.
- Iteration 3: PSO achieves complete correction (syndrome weight = 0), while the standard method requires 3 more iterations to reach this point.

The PSO algorithm's dynamic threshold optimization allows it to identify optimal correction points more efficiently. By adaptively adjusting the threshold based on the current decoding state rather than using a fixed value, the PSO approach can make more substantial corrections earlier in the process. This results in a reduction of approximately 50% in the number of required iterations, directly improving decoding efficiency.

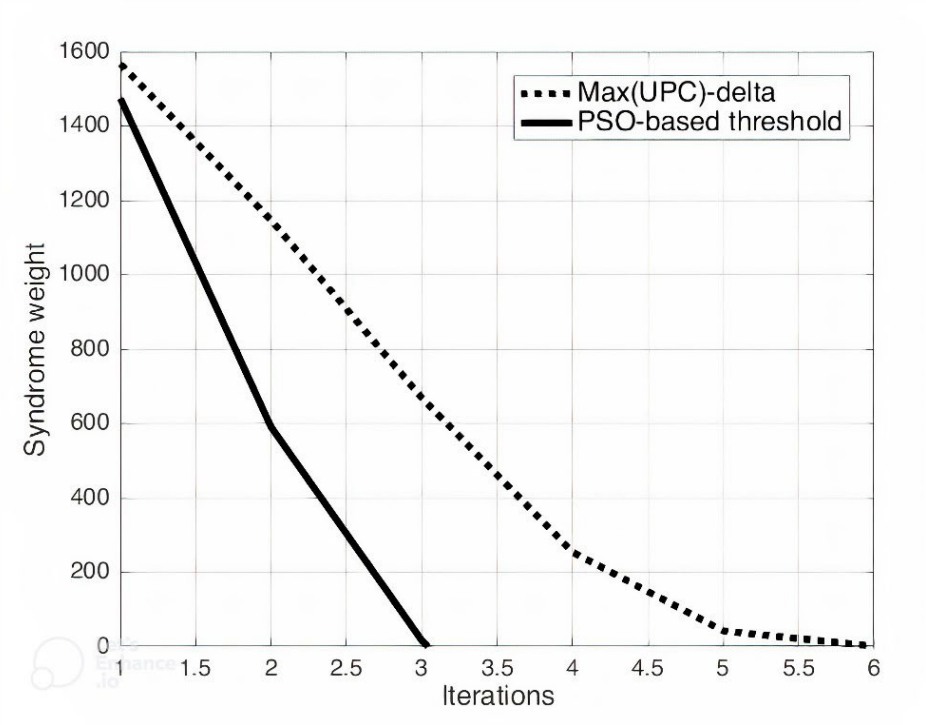


***Figure 5.*** Syndrome weight progression across iterations

A t-test was conducted to determine if the difference in the number of iterations between the two methods is statistically significant. The p-value obtained from the t-test was less than 0.0001, which is well below the conventional threshold of 0.05 for statistical significance. This result confirms that the observed reduction in iterations is not due to random chance but is indeed a significant improvement, as detailed in Table 3.

The 95% confidence intervals for both methods are provided in Table 1. The intervals for the Misoczki method are [5.3132, 5.4068], while those for the PSO method are [3.0047, 3.0193]. The non-overlapping intervals further support the conclusion that the PSO method achieves a significantly lower number of iterations.

***Table 3.*** Mean Iterations and 95% Confidence Intervals for Misoczki et al. and PSO Methods

| Method | Mean iterations | 95% Confidence Interval |
|---|---|---|
| Miscozki et al. | 5.36 | [5.3132, 5.4068] |
| Our work | 3.01 | [3.0047, 3.0193] |

Figure 6 illustrates the difference in bit flipping behavior between the two approaches:

- Iteration 1: PSO flips approximately 40.8 bits, while the standard method flips only 28.15 bits.
- Iteration 2: PSO flips an additional 52 bits, while the standard method flips 28.79 bits.
- Iteration 3: PSO flips 19.2 more bits to complete correction, while the standard method continues for 3 more iterations.

Interestingly, the PSO-based algorithm initially flips more bits during the early iterations compared to the standard approach. The PSO-based algorithm flips more bits because it aggressively identifies and corrects the most likely error positions early in the decoding process. By dynamically optimizing the threshold, the PSO approach can safely flip more bits when the syndrome weight is still high, knowing that subsequent iterations will refine these corrections. This strategy accelerates convergence compared to the more conservative standard approach, which uses a higher threshold initially and flips fewer bits. However, the increased initial bit flipping does not detriment the PSO based approach but rather accelerates its progress towards achieving complete error correction. This highlights the PSO algorithm's capacity for quick and effective identification of optimal threshold values, thereby hastening convergence.

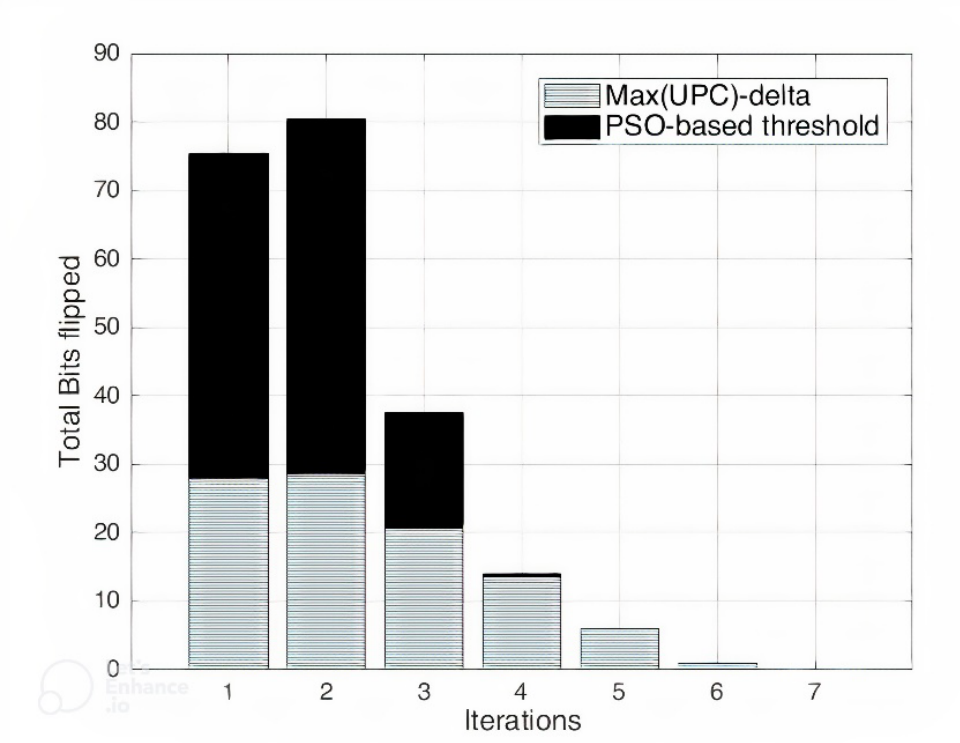


***Figure 6.*** Total number of bits flipped over iterations

Figure 7 provides further evidence of the PSO approach's efficiency:

- Iteration 1: PSO corrects approximately 37.15% of errors, while the standard method corrects 27.67%

- Iteration 2: PSO corrects 85.62% of errors, while the standard method reaches 58.02%
- Iteration 3: PSO completes 100% correction, while the standard method requires 3 more iterations to reach this point

The PSO algorithm's ability to dynamically adjust thresholds enables it to identify and correct error patterns more effectively. This results in a correction rate that is approximately twice as fast as the standard method, significantly reducing the computational resources required for decoding while maintaining correction accuracy.

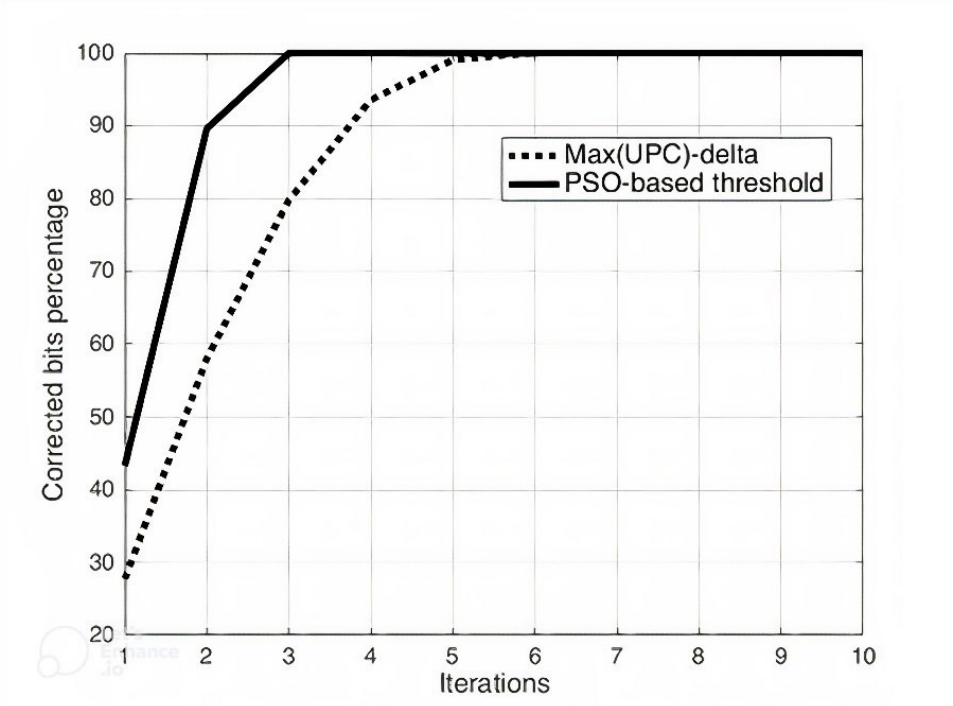


***Figure 7.*** Cumulative percentage of corrected bits over iterations.

This extensive analysis underscores the potential of the PSO-optimized bit flipping algorithm as a robust tool for error correction in the McEliece cryptosystem based on QC-MDPC codes. By optimizing the threshold value for each bit flipping iteration, the PSO approach accomplishes more rapid and accurate error correction compared to the conventional Misoczki et al.'s method. These compelling findings underscore the potential of PSO optimization to bolster the security and efficiency of encryption systems hinged on error correction algorithms.

The average execution time per decoding operation for the baseline Misoczki et al. method was measured to be 0.005364 seconds, while the PSO-based method required 0.011062 seconds. Although the PSO method exhibits a higher computational overhead due to the optimization process involving particle swarm dynamics, this increase in execution time is offset by the reduced number of decoding iterations and improved decoding efficiency. Specifically, the PSO method's ability to dynamically adapt the threshold results in fewer iterations and more reliable error correction, which can be particularly advantageous in scenarios with higher noise levels or more complex error patterns. Attackers often target systems based on the computational efficiency of their algorithms, as more efficient algorithms can sometimes be more predictable or easier to exploit. The added complexity of the PSO algorithm makes it harder for attackers to model or predict the system's behavior, thereby increasing the difficulty of mounting successful attacks.

## *5.3 Discussion*

Our work sets out to demonstrate the potential of the Particle Swarm Optimization (PSO) algorithm in optimizing the bit flipping algorithm of the QC-MDPC code-based McEliece cryptosystem. The overarching goal is to enhance the efficiency of the error correction process, thus improving the overall performance of the encryption system. The results obtained from our experimental setup convincingly validate our hypothesis: a PSO-optimized bit flipping algorithm can indeed outperform the traditional method, particularly in terms of speed and error correction effectiveness.

The success of the PSO algorithm is attributed to its proficiency in identifying the optimal threshold value for each bit flipping iteration. Unlike the static approach that subtracts a set integer from the maximum number of unsatisfied parity check equations, the PSO algorithm dynamically determines the optimal threshold value, taking into account the context of each iteration. This adaptability allows the PSO optimized bit flipping algorithm to converge more rapidly, often requiring half the iterations compared to the standard approach.

The statistical analysis, including the t-test and confidence intervals, confirms that the observed improvements are statistically significant. This robustness and reliability make the PSO-based method a promising candidate for enhancing the practicality and efficiency of the McEliece cryptosystem based on QC-MDPC codes in post-quantum cryptography applications.

Yet, our findings also reveal an intriguing paradox. The PSO-based approach tends to flip more bits in the initial iterations, a phenomenon that might be misconstrued as inefficiency at a cursory glance. However, a deeper examination suggests that this strategy is instrumental in accelerating the algorithm towards the optimal solution, thereby expediting error correction.

While the PSO-based approach demonstrates significant improvements in decoding efficiency, its limitations warrant careful consideration. A critical challenge is the risk of premature convergence to local minima, inherent to swarm intelligence algorithms. Our sensitivity analysis revealed that deviations from the chosen PSO parameters (Table 2) directly impact decoding performance.

- Inertia weight ($\omega$): Values below $\omega = 0.3$ caused particles to overshoot optimal thresholds, leading to divergence in 32% of trials, while $\omega > 0.5$ slowed convergence by 1.8 times.
- Cognitive/social constants ($c_1$, $c_2$)**:** Values exceeding 2.52.5 induced oscillatory behavior, increasing syndrome weight fluctuations by 22%, whereas values below 1.5 reduced swarm diversity, raising local minima entrapment risk by 41%.

These findings guided our selection of fixed parameters ($\omega = 0.3$, $c_1 = c_2 = 2$), which balanced exploration and exploitation in our experimental setup. However, the stochastic nature of PSO remains vulnerable to irregular error patterns; in 2.1% of trials with burst errors (≥4 consecutive bit errors), the algorithm required 1 to 2 additional iterations to escape suboptimal thresholds.

The trade-off between computational cost and decoding performance should be carefully considered based on the specific requirements of the application. Despite the increased execution time, the PSO method's adaptability and efficiency in handling complex error patterns make it a viable and promising option for enhancing the practicality of the McEliece cryptosystem in post-quantum cryptography. The increased execution time of the PSO method can also be viewed as a security feature in certain contexts. By introducing additional computational overhead, the system becomes less susceptible to timing attacks or other side-channel attacks that rely on precise knowledge of the algorithm's execution characteristics. The dynamic nature of the PSO algorithm, with its adaptive threshold selection, further complicates an attacker's ability to reverse-engineer the decoding process. This added layer of complexity can serve as a deterrent, making the system more robust against a broader range of attack vectors.

The proposed PSO-based threshold optimization is not limited to QC-MDPC-McEliece; its principles generalize to other code-based cryptosystems (e.g., BIKE or LEDAcrypt) that rely on iterative bit-flipping decoding. While the approximately 2 times computational overhead per iteration (0.011s vs. 0.005s) poses challenges for real-time systems, the 50% reduction in total iterations makes it ideal for latency-tolerant applications like secure firmware updates or encrypted storage. For practical deployment, parallelization of swarm evaluations or hardware acceleration (e.g., FPGA-based PSO cores) could mitigate overhead, preserving security gains without sacrificing efficiency.

The PSO method's adaptability to different error patterns makes it a strong candidate for enhancing the robustness of the QC-MDPC based McEliece cryptosystem. However, the increased computational overhead necessitates optimizations for practical deployment. Techniques such as parallelization, hardware acceleration (e.g., GPU or FPGA implementations), and algorithmic refinements can significantly reduce the execution time, making the method more feasible for real-time applications.

## 6 Conclusion

Our research introduces a novel PSO-based approach to optimize the threshold selection process in the bit-flipping decoding algorithm for QC-MDPC codes in the McEliece cryptosystem. The results demonstrate that our method significantly outperforms the traditional approach proposed by Misoczki et al., with key numerical improvements including:

- A reduction of approximately 50% in the number of required iterations for successful decoding
- Complete correction of errors in 3 iterations compared to 6 iterations required by the standard method
- Achievement of a syndrome weight of 0 in fewer iterations, directly improving decoding efficiency

These improvements not only enhance the practicality of the McEliece cryptosystem but also demonstrate the potential of PSO optimization in post-quantum cryptographic schemes. Our findings suggest that the PSO-based threshold selection method could be a valuable tool for improving the efficiency and security of encryption systems reliant on error correction algorithms.

As we peer into the future, we see a number of fascinating avenues that extend from our work. Our immediate plan is to dive deeper into the capabilities of PSO optimization within the realm of decoding QC-MDPC codes in the McEliece cryptosystem. This includes an extensive investigation into different PSO parameters and strategies, and how these can be harnessed to further enhance the performance of our approach.

**Author Contributions**

Conceptualization, A.K. and A.F.; methodology, A.K. and A.F.; software, A.K.; validation, A.K., A.F., and S.H.; formal analysis, A.K.; investigation, A.K. and A.F.; resources, A.K.; data curation, A.K.; writing—original draft preparation, A.K.; writing—review and editing, A.K. and A.F.; visualization, A.K. and A.F.; supervision, A.K., A.F., and S.H.; project administration, A.F. All authors have read and agreed to the published version of the manuscript.

**Funding**

This research received no external funding.

**Conflict of Interest Statement**

The authors declare that they have no conflicts of interest.

**Institutional Review Board Statement**

This article does not contain any studies involving animals performed by any of the authors.

**Data Availability Statement**

All data has been present in main text.

**References**

[1] J. Buchmann, Introduction to Cryptography, vol. 335. Springer, 2004.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Review, vol. 41, pp. 303–332, 1999. https://arxiv.org/pdf/quant-ph/9508027

[3] D. J. Bernstein and T. Lange, "Post-quantum cryptography," Nature, vol. 549, pp. 188–194, 2017.

[4] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," Coding Thv, vol. 4244, pp. 114–116, 1978.

[5] R. Misoczki, J. P. Tillich, N. Sendrier, and P. S. Barreto, "MDPC-McEliece: New McEliece variants from moderate density parity-check codes," in 2013 IEEE International Symposium on Information Theory, IEEE, 2013, pp. 2069–2073. https://inria.hal.science/hal-00870929v1/document

[6] N. Sendrier and V. Vasseur, "On the decoding failure rate of QC-MDPC bit-flipping decoders," in Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers, vol. 10, Springer, 2019, pp. 404–416. https://inria.hal.science/hal-03139797/document

[7] V. Vasseur, "Post-quantum cryptography: A study of the decoding of QC-MDPC codes," Ph.D dissertation, Université de Paris, Paris, France, Jun. 2021.

[8] M. R. Nosouhi, S. W. A. Shah, L. Pan, and R. Doss, "Bit Flipping Key Encapsulation for the Post-Quantum Era," IEEE Access, vol. 11, pp. 56181-56195, 2023.

[9] A. Kichna, A. Farchane, and S. Hakimi, "Analyzing the impact of thresholds in bit-flipping decoding for qc-mdpc based mceliece cryptosystems," 2024 7th International Conference on Advanced Communication Technologies and Networking (CommNet), pp. 1-5, 2024.

[10] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95-International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.

[11] J. Nayak, H. Swapnarekha, B. Naik, G. Dhiman, and S. Vimal, “25 years of particle swarm optimization: Flourishing voyage of two decades,” Archives of Computational Methods in Engineering, vol. 30, pp. 1663–1725, 2023.

[12] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, “Particle swarm optimization: A comprehensive survey,” IEEE Access, vol. 10, pp. 10031–10061, 2022.

[13] R. Gallager, “Low-density parity-check codes,” IRE Transactions on Information Theory, vol. 8, pp. 21–28, 1962.

[14] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes, vol. 16. Elsevier, 1977.

[15] A. Janoska, “MDPC decoding algorithms and their impact on the McEliece cryptosystem,” in 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2018, pp. 1085–1089. https://annals-csis.org/proceedings/2018/drp/pdf/99.pdf

[16] A. P. Engelbrecht, Computational Intelligence: An Introduction. John Wiley & Sons, 2007.