Solving McEliece-1409 in One Day — Cryptanalysis with the Improved BJMM Algorithm

Shintaro Narisada[1], Shusaku Uemura[1], Hiroki Okada[1,2], Hiroki Furue[2], Yusuke Aikawa[2], and Kazuhide Fukushima[1]

[1] KDDI Research, Inc, Japan
{sh-narisada,su-uemura,ir-okada,ka-fukushima}@kddi.com
[2] The University of Tokyo, Japan
furue-hiroki261@g.ecc.u-tokyo.ac.jp
aikawa@mist.i.u-tokyo.ac.jp

Abstract. Syndrome decoding problem (SDP) is the security assumption of the code-based cryptography. Three out of the four NIST-PQC round 4 candidates are code-based cryptography. Information set decoding (ISD) is known for the fastest existing algorithm to solve SDP instances with relatively high code rate. Security of code-based cryptography is often constructed on the asymptotic complexity of the ISD algorithm. However, the concrete complexity of the ISD algorithm has hardly ever been known. Recently, Esser, May and Zweydinger (Eurocrypt '22) provide the first implementation of the representation-based ISD, such as May–Meurer–Thomae (MMT) or Becker–Joux–May–Meurer (BJMM) algorithm and solve the McEliece-1284 instance in the decoding challenge, revealing the practical efficiency of these ISDs.

In this work, we propose a practically fast depth-2 BJMM algorithm and provide the first publicly available GPU implementation. We solve the McEliece-1409 instance for the first time and present concrete analysis for the record. Cryptanalysis for NIST-PQC round 4 code-based candidates against the improved BJMM algorithm is also conducted. In addition, we revise the asymptotic space complexity of the time-memory trade-off MMT algorithm presented by Esser and Zweydinger (Eurocrypt '23) from $2^{0.375n}$ to $2^{0.376n}$.

Keywords: Information Set Decoding · Representation Technique · McEliece

1 Introduction

Code-based cryptography is a public-key encryption scheme based on coding theory. Despite more than 40 years have passed since Robert McEliece developed the first code-based cryptography [27], it is receiving renewed attention today with the advent of quantum computers, as it is considered resistant to quantum attacks.

In the NIST post-quantum cryptography standardization project (NIST-PQC), three code-based cryptographic schemes — Classic McEliece [2], BIKE [3], and HQC [28] — are undergoing continuous evaluation in the fourth round [31]. Among the four submissions in this round, SIKE [5], an isogeny-based cryptography, has been deprecated due to a (classical) polynomial-time attack [13]. This situation emphasizes the urgent need for security assessment of the remaining code-based candidates.

For NIST-PQC fourth round code-based candidates, it is known that the most efficient decoding algorithm is known as Information Set Decoding (ISD). ISD is an algorithm built on the framework of Prange's algorithm [33]. Thus far, several ISD algorithms have been proposed (e.g., [6,10,15,21,25,26,36]), and their asymptotic complexity has been investigated (see Table 1). In these papers, commonly full/half distance decoding settings are considered. In the full distance setting, one computes the minimal asymptotic complexity under the weight $w = O(n)$. When $w = o(n)$, all the ISD algorithms exhibit the same asymptotic complexity $2^{cw(1+o(1))}$ with a constant c [11].

Table 1. Asymptotic time complexity $O(2^{\alpha n})$ for major ISD algorithms in full distance decoding setting. The exponent α of $O(2^{\alpha n})$ for each algorithm is listed below.

Prange [33]	Dumer [15]	MMT [25]	BJMM [6]	May-Ozerov [26]	Both-May [10]	Sieving ISD [21]
0.121	0.116	0.112	0.102	0.097	0.096	0.101

Bit security Assessments In addition to asymptotic complexity, several contributions have been made to provide security estimates for code-based cryptography [17,22,32]. In [18,20], the authors showed how to compute bit security estimates of code-based cryptography from the decoding results. Recently, Esser et al. introduced a comprehensive library for cryptographic hardness estimation [19], enabling us to estimate both bit security and the optimal parameters for a specified difficulty level of an input problem. Bernstein and Chou have developed a cryptanalysis software called CryptAttackTester, which enables detailed bit security analyses for ISD algorithms and an AES key search attack [8].

Concrete Cryptanalysis Concrete cryptanalysis is also crucial in this field. Decrypting higher-dimensional cryptography provides data points to estimates accurate security level. One known benchmark for code-based cryptography is Decoding Challenge [4].

In 2022, Esser and Zweydinger successfully solved a quasi-cyclic SDP corresponds to BIKE and HQC with parameters $n = 3138, k = 1569, w = 56$. The above authors employed the memory-optimized MMT/BJMM algorithm [20] along with the Decoding-One-Out-of-Many (DOOM) strategy [35]. In 2023, Bernstein, Lange, and Peters obtained an initial solution to a Classic McEliece-like SDP with $n = 1347, k = 1078, w = 25$. They utilized an improved variant [9] of Stern's ISD [36]. Narisada, Fukushima and Kiyomoto found a solution to the SDP for random binary linear codes for $n = 570, k = 285, w = 70$ using a GPU implementation of the MMT algorithm [30].

Recent Asymptotic Improvements Carrier et. al. provided a corrected analysis for the Both–May algorithm [12]. Additionally, Ducas et al. revealed the asymptotic complexities of the Sieving ISD [14], while Esser and Zweydinger succeeded in reducing the asymptotic space complexity of the MMT algorithm from $2^{0.053n}$ to $2^{0.0375n}$ by demonstrating its time-memory trade-offs [20].

Contributions In this study, we focus on the depth-2 BJMM algorithm, whose asymptotic time $2^{0.105n}$ is not optimal but is shown to be practical [18,20,30].

We propose an improved variant of the depth-2 BJMM algorithm, which maximizes the success probability of finding a solution. The core idea leverages multiple weight distributions of permuted solutions, which is initially proposed by Bernstein and Chou [8]. Our algorithm is different from [8] in terms of the initial list construction procedure in the enumeration phase. While our algorithm does not change the asymptotic complexity, several bits of security can be reduced from the original BJMM algorithm.

For asymptotic complexity analysis, we review the Dumer's algorithm with the Schroeppel–Shamir technique, which is initially presented in [24]. Then, we revise the asymptotic space complexity of the time-memory trade-off MMT from $2^{0.0375n}$ to $2^{0.0376n}$, which may be of independent interest.

The security of NIST-PQC fourth-round code-based candidates against existing ISD algorithms, including the improved BJMM algorithm, is also evaluated. We demonstrate that the improved BJMM algorithm exhibits the lowest bit security among existing ISD algorithms for Classic McEliece.

Furthermore, we present the first practical GPU implementation of the improved BJMM algorithm. We achieve a new record in solving the McEliece-1409 instance in the decoding challenge, which has approximately 70-bit security. Validation of our record and comparison with other state-of-the art ISD implementations are also conducted. All of our codes used in our paper are publicly available on `https://github.com/sh-narisada/CU_BJMM`.

Organization The remainder of the paper is organized as follows. Section 2 describes the notation and ISD. In Section 3, we briefly explain the depth-2 BJMM algorithm. Section 4 presents the improved depth-2 BJMM algorithm and its complexity. Section 5 conducts asymptotic complexity analyses for several ISDs using the Schroeppel–Shamir technique. In Section 6, we conduct cryptanalysis for code-based NIST-PQC round 4 candidates. Experimental results are provided in Section 7. Section 8 gives concluding remarks.

2 Preliminaries

2.1 Notation

Let $\mathbb{F}_2$ be the finite field with elements $\{0,1\}$. An n-dimensional column vector is denoted as $\mathbf{x}^\top = (x_1,\ldots,x_n) \in \mathbb{F}_2^n$ for a row vector $\mathbf{x} \in \mathbb{F}_2^{1\times n}$. Henceforth, we denote a column vector without the transposition symbol $\top$ for simplicity. A concatenation of two vectors $\mathbf{a} \in \mathbb{F}_2^m$ and $\mathbf{b} \in \mathbb{F}_2^n$ is written as $(\mathbf{a},\mathbf{b}) \in \mathbb{F}_2^{m+n}$ unless otherwise specified. Thus, we regard the tensor product of two vector spaces $\mathbb{F}_2^m \times \mathbb{F}_2^n$ as the set of concatenated vectors $\mathbb{F}_2^{m+n}$. For more than two spaces, we consider product similarly. Let the zero vector be $\mathbf{0}$. A matrix of size $m \times n$ is denoted as $\mathbf{A} \in \mathbb{F}_2^{m\times n}$. The identity matrix is represented as $\mathbf{I}$ and the zero matrix as $\mathbf{O}$. The Hamming weight for $\mathbf{x}$ is denoted by $\mathsf{wt}(\mathbf{x}) := |\{i \mid x_i = 1\}|$. Let $\mathcal{B}_w^n := \{\mathbf{x} \in \mathbb{F}_2^n \mid \mathsf{wt}(\mathbf{x}) = w\}$ be the set of all binary vectors of length n and Hamming weight w. The SDP is defined as follows.

Definition 2.1 (Syndrome Decoding Problem: SDP). Let $n,k,w \in \mathbb{N}$ such that $k \le n$ and $w \le n$. Given $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_2^{n-k}$, find a vector $\mathbf{e} \in \mathbb{F}_2^n$ of $\mathsf{wt}(\mathbf{e}) = w$ such that $\mathbf{He} = \mathbf{s}$.

This problem has been shown to be in the NP-hard class [7]. In this paper, we consider the case that an SDP has a unique solution, i.e., $\binom{n}{w} \ll 2^{n-k}$.

2.2 Information Set Decoding

ISD is a probabilistic algorithm that can be used to solve an SDP in exponential time, as originated in Prange [33]. We provide a brief overview of a common framework for ISD algorithms.

Algorithm 1 below provides the pseudo-code for an ISD algorithm. Until Line 5, column permutation and Gaussian elimination are applied to the parity-check matrix and syndrome, resulting in a systematic form $\bar{\mathbf{H}}$ and a corresponding syndrome $\bar{\mathbf{s}}$. We denote a set of all permuted solutions as $\mathcal{E}_0 = \mathcal{B}_w^n$, and a set of obtainable permuted solutions as $\mathcal{E}$, which varies across ISD algorithm. A matrix $\mathbf{P}$ is referred to as a *good permutation* when $\mathbf{Pe}$ is an element of $\mathcal{E}$. For $q := \Pr[\mathbf{P} \text{ is good}] = |\mathcal{E}|/|\mathcal{E}_0|$, we utilize a specific Search component for $(\bar{\mathbf{H}},\bar{\mathbf{s}})$, producing a permuted solution $\bar{\mathbf{e}}$ with probability q. By repeating the above procedure q^{-1} times, it is expected that one solution $\mathbf{P}\bar{\mathbf{e}}$ is obtained.

An ISD algorithm exhibits an average time complexity given by

$$q^{-1}(T_{\text{ge}} + T_{\text{search}}), \tag{1}$$

where T_{ge} is the time complexity for Gaussian elimination and T_{search} is the time complexity required for the Search component.

For instance, in the case of Prange's algorithm, the Search component checks whether $\mathsf{wt}(\bar{\mathbf{s}})$ is w or not. If it holds, the algorithm returns $\bar{\mathbf{e}} = (\bar{\mathbf{s}},\mathbf{0})$. A crucial observation regarding this algorithm is that when, fortunately, $\mathsf{wt}(\bar{\mathbf{s}}) = w$,

Algorithm 1: Information Set Decoding

Input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k}, w \in \mathbb{N}$
Output: $\mathbf{e} \in \mathbb{F}_2^n$ of weight w s.t. $\mathbf{He} = \mathbf{s}$

```
q := Pr[P is good]
repeat  /* q^{-1} times in expectation */
    Pick random permutation matrix P
    H̄ = [I_{n-k} | Ĥ] = GHP
    s̄ = Gs
    ē = Search(H̄, s̄)
    if wt(ē) = w and H̄ē = s̄ then
        return Pē
```

we have that $\bar{\mathbf{H}}(\bar{\mathbf{s}}, \mathbf{0}) = \bar{\mathbf{s}}$, which satisfies both conditions for a solution. It can be stated that $\mathcal{E} = \mathcal{B}_w^{n-k} \times \mathcal{B}_0^k$, and q is given by

$$q = \frac{\binom{n-k}{w}}{\binom{n}{w}}. \tag{2}$$

Eq. (1) is instantiated with substitutions from Eq. (2), $T_{\text{ge}} = (n-k)^2 n$ and $T_{\text{search}} = 1$.

To date, many efforts have been made to develop more efficient ISD algorithms that minimize Eq. (1) from an asymptotic perspective. However, relying solely on asymptotic analysis has resulted in a gap between theoretical results and actual time complexity.

3 Depth-2 Becker–Joux–May–Meurer Algorithm

The BJMM algorithm is a generalization of the MMT algorithm. In this paper we focus on the depth-2 variant. The inputs to the Search component in the BJMM algorithm are a semi-systematic form $\bar{\mathbf{H}}$ of the parity-check matrix and the syndrome $\bar{\mathbf{s}}$:

$$\bar{\mathbf{H}} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \mathbf{H}_1 \\ \mathbf{O} & \mathbf{H}_2 \end{pmatrix} = \mathbf{GHP}, \;\; \bar{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2) = \mathbf{Gs} \in \mathbb{F}_2^{n-k-\ell} \times \mathbb{F}_2^{\ell}, \tag{3}$$

where $\mathbf{H}_1 \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}$ and $\mathbf{H}_2 \in \mathbb{F}_2^{\ell\times(k+\ell)}$. This transformation can be achieved by applying a column permutation $\mathbf{P}$ and Gaussian elimination $\mathbf{G}$ with early abort. In the Search component, it performs the merging and filtering of several lists, each consisting of a fraction of the candidates for a solution $\bar{\mathbf{e}}$. The BJMM algorithm outputs a permuted solution $\bar{\mathbf{e}} \in \left(\mathcal{B}_{w-p'}^{n-k-\ell} \times \mathcal{B}_{p'}^{k+\ell}\right)$.

3.1 Tree-based List Construction

We describe the list construction process in the depth-2 BJMM algorithm. The output list is $L^{(0)}$ consisting of $\mathbf{z}'$, which satisfies $\mathsf{wt}(\mathbf{z}') = p' \leq 2p$ and $\mathbf{H}_2\mathbf{z}' = \mathbf{s}_2$.

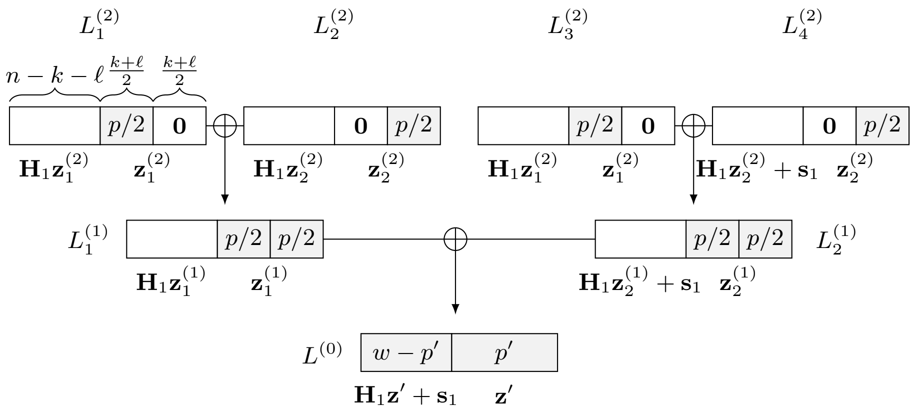

Fig. 1. Tree-based list construction of the depth-2 BJMM algorithm.

We traverse seven lists from the bottom (depth-2) to the top (depth 0), as depicted in Figure 1. First, four depth-2 base lists are prepared as follows:

$$L_1^{(2)} = L_3^{(2)} = \left\{ \mathbf{z}_1^{(2)} \in \mathbb{F}_2^{\frac{k+\ell}{2}} \times 0^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_1^{(2)}) = p/2 \right\},$$
$$L_2^{(2)} = L_4^{(2)} = \left\{ \mathbf{z}_2^{(2)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_2^{(2)}) = p/2 \right\}.$$

Then, we merge $L_1^{(2)}$ with $L_2^{(2)}$ ($L_3^{(2)}$ with $L_4^{(2)}$) to yield a depth-1 list $L_1^{(1)}$ ($L_2^{(1)}$) while filtering a pair $(\mathbf{z}_1^{(2)}, \mathbf{z}_2^{(2)})$, based on the following condition with an integer $\ell_1 \leq \ell$ and a map $\pi_{\ell_1} : \mathbb{F}_2^{\ell} \to \mathbb{F}_2^{\ell_1}, \pi_{\ell_1}(x_1, ..., x_\ell) = (x_1, ..., x_{\ell_1})$:

$$L_1^{(1)} = \left\{ \mathbf{z}_1^{(1)} \;\middle|\; \mathbf{z}_1^{(2)} \in L_1^{(2)}, \mathbf{z}_2^{(2)} \in L_2^{(2)}, \mathbf{z}_1^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)}, \pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)}) = \mathbf{t} \right\},$$
$$L_2^{(1)} = \left\{ \mathbf{z}_2^{(1)} \;\middle|\; \mathbf{z}_1^{(2)} \in L_3^{(2)}, \mathbf{z}_2^{(2)} \in L_4^{(2)}, \mathbf{z}_2^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)}, \pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_2^{(1)} + \mathbf{s}_2) = \mathbf{t} \right\}.$$

Here, $\mathbf{t} \in \mathbb{F}_2^{\ell_1}$ is a randomly chosen vector. Note that $\mathsf{wt}(\mathbf{z}_1^{(1)}) = \mathsf{wt}(\mathbf{z}_2^{(1)}) = p$, since there is no overlap at the 1's position between $\mathbf{z}_1^{(2)}$ and $\mathbf{z}_2^{(2)}$. Then, $L_1^{(1)}$ and $L_2^{(1)}$ are merged under a specific condition to yield a list $L^{(0)}$:

$$L^{(0)} = \left\{ \mathbf{z}' = \mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)} \;\middle|\; \mathbf{z}_1^{(1)} \in L_1^{(1)}, \mathbf{z}_2^{(1)} \in L_2^{(1)}, \mathbf{z}', \mathbf{H}_2\mathbf{z}' = \mathbf{s}_2, \mathsf{wt}(\mathbf{z}') = p' \right\}. \tag{4}$$

Since we already have that $\pi_{\ell_1}(\mathbf{H}_2(\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)})) = \pi_{\ell_1}(\mathbf{s}_2)$ and $\mathsf{wt}(\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)}) \leq 2p$, we can obtain $L^{(0)}$ by checking the remaining $\ell - \ell_1$ indices of $\mathbf{H}_2(\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)})$ and Hamming weight of $\mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)}$. Now, if we set $\mathbf{z}'' = \mathbf{H}_1\mathbf{z}' + \mathbf{s}_1$, then the first condition of a solution, $\bar{\mathbf{H}}(\mathbf{z}'', \mathbf{z}') = (\mathbf{s}_1, \mathbf{H}_2\mathbf{z}') = \bar{\mathbf{s}}$ is satisfied. We need to verify that $(\mathbf{z}'', \mathbf{z}')$ has the desired weight distribution, i.e., $(\mathbf{z}'', \mathbf{z}') \in \left(\mathcal{B}_{w-p'}^{n-k-\ell} \times \mathcal{B}_{p'}^{k+\ell}\right)$. When $\mathsf{wt}(\mathbf{z}'') = w - p'$, we observe that $\mathsf{wt}(\bar{\mathbf{z}}) = w$ for $\bar{\mathbf{z}} = (\mathbf{z}'', \mathbf{z}')$. Thus, $\mathbf{P}\bar{\mathbf{z}}$ is the solution to the SDP.

3.2 Computational Complexity

We review the complexity analysis of the depth-2 BJMM algorithm. The time complexity per iteration of **repeat** in Algorithm 1 is dominated by the time complexity of Gaussian elimination $T_{\mathrm{ge}} = (n-k)^2 n$ and T_{search} required for the SEARCH component. T_{search} is the sum of the time complexities for base list construction and for the merging of lists at each depth. For $|L^{(2)}| = \binom{(\ell+k)/2}{p/2}$ and $|L^{(1)}| = \max(1, 2^{-\ell_1}|L^{(2)}|^2)$, we obtain that

$$T_{\mathrm{search}} = 2|L^{(2)}| + 2\max(|L^{(2)}|, 2^{-\ell_1}|L^{(2)}|^2 + \max(|L^{(1)}|, 2^{-\ell+\ell_1}|L^{(1)}|^2). \quad (5)$$

Note that we do not need space for $L^{(0)}$ since we can enumerate $\mathbf{z}'$ on-the-fly from $L_1^{(1)}$ and $L_2^{(1)}$. The set of obtainable permuted solutions is $\mathcal{E} = \mathcal{B}_{w-p'}^{n-k-\ell} \times \mathcal{B}_{p'}^{k+\ell}$. The success probability q is given by

$$q = \frac{\binom{n-k-\ell}{w-p'}\binom{k+\ell}{p'}}{\binom{n}{w}}. \quad (6)$$

The average time complexity is Eq. (1) with $T_{\mathrm{ge}} = (n-k)^2 n$, Eq. (5) and Eq. (6). The space complexity of the BJMM algorithm is

$$(n-k)n + 2|L^{(2)}| + 2\max(1, 2^{-\ell_1}|L^{(2)}|^2). \quad (7)$$

In practice, we search for a valid integer parameter set (p, ℓ, ℓ_1) to minimize the time complexity. To efficiently find it, several estimators have been proposed (e.g., [17,19]). The parameter ℓ_1 must be chosen carefully as it is related to the *representations*.

A (split) representation of a weight-ω_1 vector $\mathbf{z} \in \mathbb{F}_2^n$ is a pair of vectors $(\mathbf{z}_1, \mathbf{z}_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, satisfying $\mathbf{z} = \mathbf{z}_1 + \mathbf{z}_2$ and $\mathsf{wt}(\mathbf{z}_1) = \mathsf{wt}(\mathbf{z}_2) = \omega_2 \geq \omega_1/2$. In the BJMM algorithm, the number of representations for a weight-p' $\mathbf{z}'$ as a sum of two weight-p vectors $\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}$ is

$$R = \binom{p'}{p'/2}\binom{k+\ell-p'}{p-p'/2}. \quad (8)$$

In [25], valid parameters are searched under the condition that at least a single representation of a solution is expected to be contained in $L^{(0)}$, i.e., $\ell_1 \leq \log_2 R$. In [20], the authors consider the case where $\ell_1 > \log_2 R$. When $\ell_1 > \log_2 R$, the probability ρ_{repr} of at least one representation being included in $L^{(0)}$ is given by

$$\rho_{\mathrm{repr}} := 1 - (1 - 2^{-\ell_1})^R \approx 2^{-\ell_1} R. \quad (9)$$

They demonstrate that the decrease in the number of representations can be compensated by repeating the SEARCH component ${\rho_{\mathrm{repr}}}^{-1}$ times. The time complexity for the BJMM algorithm in the case of $\ell_1 > \log_2 R$ is given by

$$q^{-1}(T_{\mathrm{ge}} + {\rho_{\mathrm{repr}}}^{-1}\, T_{\mathrm{search}}).$$

The advantage of setting $\ell_1 > \log_2 R$ is to show a time-memory trade-off for cases where $\ell_1 \leq \log_2 R$, which implies that a portion of the space complexity required in the SEARCH component can be offset by additional time complexity. Adopting a relatively large ℓ_1 can also result in practical reductions in actual runtime, as indicated in [20,30].

4 Improved Depth-2 BJMM Algorithm

In this section, we present the improved depth-2 BJMM algorithm and its concrete complexity analysis.

4.1 Algorithm Detail

Algorithm 2: IMPROVED DEPTH-2 BJMM

Input: $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_2^{n-k}, w \in \mathbb{N}$

Output: $\mathbf{e} \in \mathbb{F}_2^n$ of weight w s.t. $\mathbf{He} = \mathbf{s}$

1 Choose optimal ℓ, ℓ_1, p

2 **repeat**

3 Pick random permutation matrix $\mathbf{P}$

4 $\bar{\mathbf{H}} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \mathbf{H}_1 \\ \mathbf{O} & \mathbf{H}_2 \end{pmatrix} = \mathbf{GHP}$

5 $\bar{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2) = \mathbf{Gs}$

6 Compute

$$\bar{L}_1^{(2)} = \bar{L}_3^{(2)} = \{\mathbf{z}_1^{(2)} \in \mathbb{F}_2^{\frac{k+\ell}{2}} \times 0^{\frac{k+\ell}{2}} \mid 0 \leq \mathsf{wt}(\mathbf{z}_1^{(2)}) \leq p/2\}$$
$$\bar{L}_2^{(2)} = \bar{L}_4^{(2)} = \{\mathbf{z}_2^{(2)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{k+\ell}{2}} \mid 0 \leq \mathsf{wt}(\mathbf{z}_2^{(2)}) \leq p/2\}$$

7 Compute

$$\bar{L}_1^{(1)} = \{\mathbf{z}_1^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)} \mid \pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)}) = \mathbf{t}\} \text{ from } \bar{L}_1^{(2)} \text{ and } \bar{L}_2^{(2)}$$
$$\bar{L}_2^{(1)} = \{\mathbf{z}_2^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)} \mid \pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)} + \mathbf{s}_2) = \mathbf{t}\} \text{ from } \bar{L}_3^{(2)} \text{ and } \bar{L}_4^{(2)}$$

8 Compute $\bar{L}^{(0)} = \{\mathbf{z}' = \mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)} \mid \mathbf{H}_2\mathbf{z}' = \mathbf{s}_2\}$ from $\bar{L}_1^{(1)}$ and $\bar{L}_2^{(1)}$

9 **for** $\mathbf{z}' \in \bar{L}^{(0)}$ **do**

10 $\bar{\mathbf{z}} = (\mathbf{H}_1\mathbf{z}' + \mathbf{s}_1, \mathbf{z}')$

11 **if** $\mathsf{wt}(\bar{\mathbf{z}}) = w$ **then**

12 **return** $\mathbf{P}\bar{\mathbf{z}}$

Algorithm 2 describes the pseudo-code of the improved depth-2 BJMM algorithm. Line 6 and 11 in Algorithm 2 are different compared with the standard depth-2 BJMM algorithm. In Line 6, we enumerate all vectors whose weights are less than or equal to $p/2$, instead of specifically enumerating weight-$p/2$ vectors,

$$\bar{L}_1^{(2)} = \bar{L}_3^{(2)} = \left\{ \mathbf{z}_1^{(2)} \in \mathbb{F}_2^{\frac{k+\ell}{2}} \times 0^{\frac{k+\ell}{2}} \;\middle|\; 0 \le \mathsf{wt}(\mathbf{z}_1^{(2)}) \le p/2 \right\},$$
$$\bar{L}_2^{(2)} = \bar{L}_4^{(2)} = \left\{ \mathbf{z}_2^{(2)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{k+\ell}{2}} \;\middle|\; 0 \le \mathsf{wt}(\mathbf{z}_2^{(2)}) \le p/2 \right\}.$$

This approach leads to an slight increase in the base list size,

$$|\bar{L}^{(2)}| = \sum_{0 \le i \le p/2} \binom{(k+\ell)/2}{p/2 - i}.$$

However, this also increases the number of permuted solutions in the final list $\bar{L}^{(0)}$. In Line 11, we check $\mathsf{wt}(\bar{\mathbf{z}}) = w$ instead of the original way $\mathsf{wt}(\mathbf{H}_1\mathbf{z}' + \mathbf{s}_1) = w - 2p$, as suggested in [8]. This corresponds to removing the constraint on the weight distribution of the solution. In [8], the authors showed that this replacement increases the success probability to find the solution with no additional cost. We show that combining Line 6 with Line 11 maximizes the success probability with a slight increase in computational complexity.

4.2 Concrete Complexity Analysis

To obtain the concrete complexity of the improved BJMM algorithm, we first show the probability that a permuted solution with a specific weight distribution is included in the final list $\bar{L}^{(0)}$.

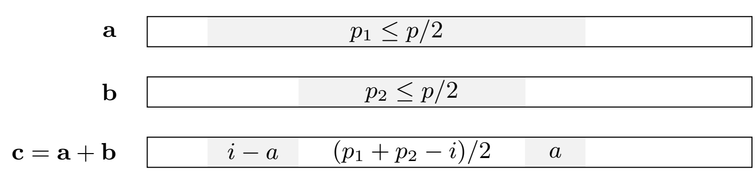

Fig. 2. An example of $\mathbb{F}_2$-addition that yields a weight-i vector $\mathbf{c}$ from a weight-p_1 vector $\mathbf{a}$ and a weight-p_2 vector $\mathbf{b}$. We have $(p_1 + p_2 - i)/2$ positions of 1's duplicated between $\mathbf{a}$ and $\mathbf{b}$. In this example, we have $i - a$ ones on the left side of $\mathbf{c}$ and a ones on the right side.

Proposition 4.1. *Let $i, j \in \mathbb{N}$ such that $0 \le i, j \le p$. Assuming that $\boldsymbol{P}$ permutes the solution as $\boldsymbol{P}\boldsymbol{e} = (\boldsymbol{e}'', \boldsymbol{e}')$ so that $\boldsymbol{e}'' \in \mathcal{B}_{w-i-j}^{n-k-\ell}$ and $\boldsymbol{e}' \in \left(\mathcal{B}_i^{(k+\ell)/2} \times \mathcal{B}_j^{(k+\ell)/2}\right)$, Then, we say*

$$\Pr\left[\boldsymbol{e}' \in \bar{L}^{(0)}\right] = \rho_{i,j}, \tag{10}$$

where

$$\rho_{i,j} := \begin{cases} 1-(1-2^{-2\ell_1})^{R_0^2} & (i=j=0), \\ 1-(1-2^{-\ell_1})^{R_iR_j} & (\textit{otherwise}), \end{cases}$$

$$R_i := \sum_{(p_1,p_2)\in\mathcal{P}_i} \binom{i}{\lfloor i/2 \rfloor}\binom{(k+\ell)/2-i}{(p_1+p_2-i)/2}, \tag{11}$$

$$\mathcal{P}_i := \left\{(p_1,p_2) \;\middle|\; p_1,p_2 \le \frac{p}{2}, |p_1-p_2| \le i \le p_1+p_2, p_1+p_2 \equiv i \bmod 2\right\}. \tag{12}$$

Proof. First, we derive a set $\mathcal{P}_i$, which consists of feasible weight splits (p_1,p_2) of i, i.e., we enumerate possible weight pair $(\mathsf{wt}(\mathbf{a}),\mathsf{wt}(\mathbf{b}))$ for $\mathbb{F}_2$-addition $\mathbf{c}=\mathbf{a}+\mathbf{b}$ with $\mathsf{wt}(\mathbf{c})=i$, where $\mathsf{wt}(\mathbf{a}) \le p/2$ and $\mathsf{wt}(\mathbf{b}) \le p/2$. Let ϵ be the number of duplicates in 1's position between $\mathbf{a}$ and $\mathbf{b}$. Then, we can enumerate (p_1,p_2) satisfying $i=p_1+p_2-2\epsilon$ for $0 \le 2\epsilon \le \min(p_1,p_2)$, which corresponds to $\mathcal{P}_i$.

We can count the number of representations for $\mathbf{e}'=\mathbf{z}_1^{(1)}+\mathbf{z}_2^{(1)}$. Let R_{i,p_1,p_2} be the number of representations of a vector $\mathbf{a}=\mathbf{b}+\mathbf{c}$, where $\mathbf{a}\in\mathcal{B}_i^{(k+\ell)/2}$, $\mathbf{b}\in\mathcal{B}_{p_1}^{(k+\ell)/2}$ and $\mathbf{c}\in\mathcal{B}_{p_2}^{(k+\ell)/2}$, as depicted in Figure 2. The set of 1-coordinates in $\mathbf{a}$ can be split in $\binom{i}{\lfloor i/2 \rfloor}$ ways as $1=1+0$ or $1=0+1$, where $\lfloor\cdot\rfloor$ is required to account for the case where i is an odd integer. For each split representation, the set of 0-coordinates can be split in $\binom{(k+\ell)/2-i}{(p_1+p_2-i)/2}$ ways by $0=1+1$. In total, $\mathbf{a}$ has $R_{i,p_1,p_2}=\binom{i}{\lfloor i/2 \rfloor}\binom{(k+\ell)/2-i}{(p_1+p_2-i)/2}$ representations. Hence, for $\mathbf{e}'=\mathbf{z}_1^{(1)}+\mathbf{z}_2^{(1)}$ we have $\sum_{(p_1,p_2)\in\mathcal{P}_i} R_{i,p_i,p_2}\sum_{(p_1,p_2)\in\mathcal{P}_j} R_{j,p_1,p_2}$ representations, which corresponds to R_iR_j in Eq. (11).

When $\mathbf{e}'\neq\mathbf{0}$ for a representation $\mathbf{e}'=\mathbf{z}_1^{(1)}+\mathbf{z}_2^{(1)}$ satisfying $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})=\mathbf{t}$, then $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_2^{(1)}+\mathbf{s}_2)=\mathbf{t}$ automatically holds. Since the probability of $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})=\mathbf{t}$ is $2^{-\ell_1}$ and we have R_iR_j representations for $\mathbf{e}'$, Eq. (10) is $1-(1-2^{-\ell_1})^{R_iR_j}$. Note that each representation has a random value for $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})$ and we consider the probability that at least one out of R_iR_j representations satisfies $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})=\mathbf{t}$. When $\mathbf{e}'=\mathbf{0}$, both $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})=\mathbf{t}$ and $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_2^{(1)}+\mathbf{s}_2)=\mathbf{t}$ hold only when $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})=\mathbf{t}$ and $\pi_{\ell_1}(\mathbf{s}_2)=\mathbf{t}$ hold. Hence, Eq. (10) is $1-(1-2^{-2\ell_1})^{R_0^2}$. □

From the above proposition, we can derive the concrete complexity of the BJMM algorithm. The set of obtainable permuted solutions $\mathcal{E}$ for the proposed algorithm is given by

$$\mathcal{E}=\bigcup_{0\le i,j\le p}\mathcal{C}_{i,j},$$

where $\mathcal{C}_{i,j}:=\mathcal{B}_{w-i-j}^{n-k-\ell}\times\mathcal{B}_i^{(k+\ell)/2}\times\mathcal{B}_j^{(k+\ell)/2}$ and $|\mathcal{C}_{i,j}|=\binom{n-k-\ell}{w-i-j}\binom{(k+\ell)/2}{i}\binom{(k+\ell)/2}{j}$. From Proposition 4.1, we expect to have

$$|\mathcal{C}_{i,j}|\cdot\Pr\left[\mathbf{e}'\in\bar{L}^{(0)} \;\middle|\; \mathbf{e}'\in\left(\mathcal{B}_i^{(k+\ell)/2}\times\mathcal{B}_j^{(k+\ell)/2}\right)\right]$$

obtainable permuted solutions for each pair (i, j). Hence, $q := \Pr[\mathbf{P} \text{ is good}]$ is given by

$$q = \frac{\sum_{0 \le i,j \le p} |\mathcal{C}_{i,j}| \rho_{i,j}}{\binom{n}{w}}. \tag{13}$$

For the time complexity, we obtain

$$T_{\text{search}} = 2|\bar{L}^{(2)}| + 2\max(|\bar{L}^{(2)}|, 2^{-\ell_1}|\bar{L}^{(2)}|^2) + \max(|\bar{L}^{(1)}|, 2^{-\ell+\ell_1}|\bar{L}^{(1)}|^2), \tag{14}$$

where, $|\bar{L}^{(2)}| = \sum_{0 \le i \le p/2} \binom{(\ell+k)/2}{p/2-i}$ and $|\bar{L}^{(1)}| = \max(1, 2^{-\ell_1}|\bar{L}^{(2)}|^2)$. The average time complexity of the improved BJMM algorithm is described by Eq. (1) with Eq. (13), Eq. (14) and $T_{\text{ge}} = (n-k)^2 n$. The space complexity of the algorithm is

$$(n-k)n + 2|\bar{L}^{(2)}| + 2\max(1, 2^{-\ell_1}|\bar{L}^{(2)}|^2). \tag{15}$$

From Eq. (14) and Eq. (15), the increase ratios for both time and space complexities from original BJMM are dominated by $|\bar{L}^{(2)}|$, given that $\ell_1 \approx \log_2 |\bar{L}^{(2)}|$. The increase ratio of the base list is

$$\frac{|\bar{L}^{(2)}|}{|L^{(2)}|} = 1 + \frac{p}{k+\ell-p+2} + O(p^2 k^{-2}).$$

Since $p \ll k$, it is dominated by pk^{-1}, which is negligible for large k.

The improved BJMM algorithm also provides time-memory trade-offs by selecting a large ℓ_1. A larger ℓ_1 reduces both the space complexity and the expected number of obtainable solutions $\sum_{0 \le i,j \le p} |\mathcal{C}_{i,j}| \rho_{i,j}$, which is directly compensated for by increasing the number of outer loops.

5 Asymptotic Analysis for Schroeppel–Shamir ISD

This section provides asymptotic complexity analysis of Dumer's algorithm and the time-memory trade-off MMT algorithm with the Schroeppel–Shamir technique.

5.1 The Schroeppel–Shamir Technique

The Schroeppel–Shamir technique [34] can reduce the memory complexity of a standard meet-in-the-middle (MITM) attack for the 2-list matching problem. Assume that we aim to find a pair $(\mathbf{x}_1, \mathbf{x}_2) \in L_1 \times L_2$ such that $\mathbf{x}_1 = \mathbf{x}_2$ on certain ℓ coordinates. This can be solved by the MITM in time $O(|D|)$ and memory $O(|D|)$, where $|D| = |L_1| = |L_2|$ and $\ell = \log_2 |D|$. When employing the Schroeppel–Shamir technique, it is known that this problem can be solved with the same time complexity and reduced memory complexity of $O(|D|^{1/2})$.

The algorithm decomposes $L_1 = L_{1,1} \times L_{1,2}$ and $L_2 = L_{2,1} \times L_{2,2}$, where $|L_{i,j}| = |D|^{1/2}$. We set $r = \log_2 |D|^{1/2} = \ell/2$. Then, we create a list $\tilde{L}_1$ from $L_{1,1}$

and $L_{1,2}$, which is a 2^{-r}-fraction of L_1 consisting of $\mathbf{x}_1 = \mathbf{x}_{1,1}+\mathbf{x}_{1,2}$ s.t. $\pi_r(\mathbf{x}_{1,1})+\pi_r(\mathbf{x}_{1,2}) = \mathbf{t}$ for some $\mathbf{t} \in \mathbb{F}_2^r$, where $\mathbf{x}_{1,1} \in L_{1,1}$ and $\mathbf{x}_{1,2} \in L_{1,2}$. The list $\tilde{L}_1$ is constructed in time and memory of $|D|^{1/2}$. Analogously, $\tilde{L}_2$ is constructed from $L_{2,1}$ and $L_{2,2}$ consisting of $\mathbf{x}_2 = \mathbf{x}_{2,1} + \mathbf{x}_{2,2}$, s.t. $\pi_r(\mathbf{x}_{2,1}) + \pi_r(\mathbf{x}_{2,2}) = \mathbf{t}$. We obtain a 2^{-r}-fraction of solution pairs in time $|\tilde{L}_1||\tilde{L}_2|/2^{\ell-r} = |D|^{1/2}$ and memory $|D|^{1/2}$. Note that for all pairs $(\mathbf{x}_1, \mathbf{x}_2) \in \tilde{L}_1 \times \tilde{L}_2$, $\mathbf{x}_1 = \mathbf{x}_2$ is satisfied on r coordinates. Therefore, we need to find a pair matching the remaining $\ell - r = \ell/2$ coordinates. The above procedure is iterated $|D|^{1/2}$ times for all $\mathbf{t} \in \mathbb{F}_2^r$. In total, we obtain all solution pairs $(\mathbf{x}_1, \mathbf{x}_2) \in L_1 \times L_2$ in time $O(|D|^{1/2}|D|^{1/2}) = O(D)$ and memory $O(|D|^{1/2})$.

5.2 Dumer's Algorithm with Schroeppel–Shamir Technique

We review the asymptotic complexity analysis of Dumer's algorithm shown in [24] and provide a numerical optimization result. Assume we have the semi-systematic form $\bar{\mathbf{H}}$ and the syndrome $\bar{\mathbf{s}}$ as shown in Eq. (3). In Dumer's algorithm, we aim to find a permuted solution $\bar{\mathbf{e}} \in \left(\mathcal{B}_{w-2p}^{n-k-\ell} \times \mathcal{B}_p^{(k+\ell)/2} \times \mathcal{B}_p^{(k+\ell)/2}\right)$. To do so, we construct two base lists as follows:

$$L_1^{(1)} = \left\{\mathbf{z}_1^{(1)} \in \mathbb{F}_2^{\frac{k+\ell}{2}} \times 0^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_1^{(1)}) = p\right\},$$
$$L_2^{(1)} = \left\{\mathbf{z}_2^{(1)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_2^{(1)}) = p\right\}.$$

We then enumerate all pairs $(\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}) \in L_1^{(1)} \times L_2^{(1)}$ s.t. $\mathbf{H}_2\mathbf{z}' = \mathbf{s}_2$, where $\mathbf{z}' = \mathbf{z}_1^{(1)} + \mathbf{z}_2^{(1)}$. For $\mathbf{z}'' = \mathbf{H}_1\mathbf{z}' + \mathbf{s}_1$, if $\mathsf{wt}(\mathbf{z}'') = w - 2p$, then, $\mathbf{P}(\mathbf{z}'', \mathbf{z}')$ is the solution. There is no representation in Dumer's algorithm. The asymptotic time complexity of Dumer's algorithm is

$$q^{-1} \max(|D|, 2^{-\ell}|D|^2), \tag{16}$$

where $q = \binom{k+\ell}{2p}\binom{n-k-\ell}{w-2p}\binom{n}{w}^{-1}$ and $|D| = \binom{(k+\ell)/2}{p}$. For readability, we omit Stirling's approximation from the asymptotic complexity. We employ the approximation $\binom{k+\ell}{2p} \approx \binom{(k+\ell)/2}{p}^2$, which is equivalent asymptotically. The space complexity is $|D|$ when we ignore polynomial factors.

One can apply the Schroeppel–Shamir technique in Dumer's algorithm by decomposing $L_1^{(1)} = L_1^{(2)} \times L_2^{(2)}$ and $L_2^{(1)} = L_3^{(2)} \times L_4^{(2)}$:

$$L_1^{(2)} = \left\{\mathbf{z}_1^{(2)} \in \mathbb{F}_2^{\frac{k+\ell}{4}} \times 0^{\frac{3(k+\ell)}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_1^{(2)}) = p/2\right\},$$
$$L_2^{(2)} = \left\{\mathbf{z}_2^{(2)} \in 0^{\frac{k+\ell}{4}} \times \mathbb{F}_2^{\frac{(k+\ell)}{4}} \times 0^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_2^{(2)}) = p/2\right\},$$
$$L_3^{(2)} = \left\{\mathbf{z}_3^{(2)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{(k+\ell)}{4}} \times 0^{\frac{k+\ell}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_3^{(2)}) = p/2\right\},$$
$$L_4^{(2)} = \left\{\mathbf{z}_4^{(2)} \in 0^{\frac{3(k+\ell)}{4}} \times \mathbb{F}_2^{\frac{k+\ell}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_4^{(2)}) = p/2\right\}.$$

We create a 2^{-r}-fraction list $\tilde{L}_1^{(2)} \subset L_1^{(2)} \times L_2^{(2)}$, whose element is $\mathbf{z}_1^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)}$, s.t. $\pi_r(\mathbf{H}_2\mathbf{z}_1^{(2)}) + \pi_r(\mathbf{H}_2\mathbf{z}_2^{(2)}) = \mathbf{t}$, where $r = |D|^{1/2} \leq \ell$ is a parameter and $\mathbf{t} \in \mathbb{F}_2^r$ is some vector. The time and memory complexities required to construct $\tilde{L}_1^{(1)}$ are $|D|^{1/2}$. Analogously, we create a 2^{-r}-fraction list $\tilde{L}_2^{(1)} \subset L_3^{(2)} \times L_4^{(2)}$, whose element is $\mathbf{z}_2^{(1)} = \mathbf{z}_3^{(2)} + \mathbf{z}_4^{(2)}$, s.t. $\pi_r(\mathbf{H}_2\mathbf{z}_3^{(2)}) + \pi_r(\mathbf{H}_2\mathbf{z}_4^{(2)}) = \pi_r(\mathbf{s}_2) + \mathbf{t}$.

We want to find 2^{-r}-fraction of ℓ-matched pairs $(\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}) \in L_1^{(1)} \times L_2^{(1)}$ s.t. $\mathbf{H}_2\mathbf{z}_1^{(1)} + \mathbf{H}_2\mathbf{z}_2^{(1)} = \mathbf{s}_2$. Since we already have a 2^{-r}-fraction of r-matched pairs $(\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}) \in \tilde{L}_1^{(1)} \times \tilde{L}_2^{(1)}$ s.t. $\pi_r(\mathbf{H}_2\mathbf{z}_1^{(1)}) + \pi_r(\mathbf{H}_2\mathbf{z}_2^{(1)}) = \pi_r(\mathbf{s}_2)$, this can be obtained in time $\max(|D|^{1/2}, 2^{r-\ell}|D|^{1/2}|D|^{1/2})$ and memory $|D|^{1/2}$. We iterate the above procedure for all $\mathbf{t}$. Therefore, the asymptotic time complexity is

$$q^{-1}2^r \max(|D|^{1/2}, 2^{r-\ell}|D|). \tag{17}$$

When $\ell \geq r/2$, Eq. (17) is equivalent to Eq. (16). The space complexity is reduced to $|D|^{1/2}$.

Numerical Optimization We implement Dumer's ISD with the Schroeppel–Shamir technique using the library developed by Esser[3] [16], and perform numerical optimization in the full distance decoding setting. In the optimization, binomial coefficients are approximated by Stirling's approximation. For each parameters o_i used in ISD algorithms, let $o_i = \tilde{o}_i \cdot n$, where $0 \leq \tilde{o}_i \leq 1$. We denote $\tilde{k} = k/n$ as the code rate. During optimization, we search for parameters that yield minimal time complexity $T_{\min}^{\tilde{k}}$ for each code rate $\tilde{k}$. In full distance decoding, we set $\tilde{w} = H^{-1}(1-\tilde{k})$. The asymptotic time complexity is $\max_{\tilde{k}} T_{\min}^{\tilde{k}}$.

The asymptotic complexity of Dumer's algorithm is

$$T = 2^{0.116n} \text{ and } S = 2^{0.0177n},$$

at $\tilde{k} = 0.43$ and $\tilde{w} = 0.1273$, with optimal parameters of

$$\tilde{p} = 0.005088,\ \tilde{r} = 0.01766,\ \tilde{\ell} = 0.03532.$$

The Schroeppel–Shamir technique reduces the asymptotic space complexity from the original value $S = 2^{0.0353n}$ to its square root $S = 2^{0.0177n}$ while maintaining the same time complexity, where $r = |L^{(2)}|$ and $\ell = 2r$.

5.3 MMT Algorithm with Schroeppel–Shamir Technique

We provide an analysis of the time-memory trade-off MMT algorithm with the Schroeppel–Shamir technique, which is initially introduced in [20]. First, we create depth-1 lists $L_1^{(1)}$ and $L_2^{(1)}$ via Schroeppel–Shamir. Since $\mathbf{z}_1^{(1)}$ is constructed from a pair $(\mathbf{z}_1^{(2)}, \mathbf{z}_2^{(2)})$, where $\mathbf{z}_1^{(1)} = \mathbf{z}_1^{(2)} + \mathbf{z}_2^{(2)}$, we can consider a 2^{-r} fraction of the set of pairs by imposing $\pi_r(\mathbf{H}_2\mathbf{z}_1^{(2)}) = \mathbf{t}_1$ and $\pi_r(\mathbf{H}_2\mathbf{z}_2^{(2)}) = \mathbf{t}_1$ for $r \leq \ell_1$

[3] Available at `https://github.com/Memphisd/Revisiting-NN-ISD`.

and some $\mathbf{t}_1 \in \mathbb{F}_2^r$. Analogously, $\mathbf{z}_2^{(1)}$ is formed from a pair $(\mathbf{z}_3^{(2)}, \mathbf{z}_4^{(2)})$, where $\mathbf{z}_2^{(1)} = \mathbf{z}_3^{(2)} + \mathbf{z}_4^{(2)}$. A 2^{-r} fraction is obtained by imposing $\pi_r(\mathbf{H}_2\mathbf{z}_3^{(2)}) = \mathbf{t}_2$ and $\pi_r(\mathbf{H}_2\mathbf{z}_4^{(2)}) = \pi_r(\mathbf{s}_2) + \mathbf{t}_2$ for some $\mathbf{t}_2 \in \mathbb{F}_2^r$. There are 2^{2r} combinations for a pair $(\mathbf{t}_1, \mathbf{t}_2)$, as $\mathbf{t}_1$ is independent of $\mathbf{t}_2$. Therefore, we have 2^{-2r} fraction of $(\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}) \in L_1^{(1)} \times L_2^{(1)}$ for some pair $(\mathbf{t}_1, \mathbf{t}_2)$, i.e., for depth-2 Schroeppel–Shamir, we require 2^{-2r} iterations as compensation for reducing the list size to 2^{-r}.

Concretely, we first create depth-3 lists by decomposing $L_i^{(2)} = L_{2i-1}^{(3)} \times L_{2i}^{(3)}$ for $1 \le i \le 4$.

$$
\begin{aligned}
L_1^{(3)} = L_5^{(3)} &= \left\{ \mathbf{z}_1^{(3)} \in \mathbb{F}_2^{\frac{k+\ell}{4}} \times 0^{\frac{3(k+\ell)}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_1^{(3)}) = p/4 \right\}, \\
L_2^{(3)} = L_6^{(3)} &= \left\{ \mathbf{z}_2^{(3)} \in 0^{\frac{k+\ell}{4}} \times \mathbb{F}_2^{\frac{(k+\ell)}{4}} \times 0^{\frac{k+\ell}{2}} \;\middle|\; \mathsf{wt}(\mathbf{z}_2^{(3)}) = p/4 \right\}, \\
L_3^{(3)} = L_7^{(3)} &= \left\{ \mathbf{z}_3^{(3)} \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{(k+\ell)}{4}} \times 0^{\frac{k+\ell}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_3^{(3)}) = p/4 \right\}, \\
L_4^{(3)} = L_8^{(3)} &= \left\{ \mathbf{z}_4^{(3)} \in 0^{\frac{3(k+\ell)}{4}} \times \mathbb{F}_2^{\frac{k+\ell}{4}} \;\middle|\; \mathsf{wt}(\mathbf{z}_4^{(3)}) = p/4 \right\},
\end{aligned}
$$

where $|L_i^{(3)}| = \binom{(k+\ell)/4}{p/4} \approx |D|^{1/2}$ for $|D| = \binom{(k+\ell)/2}{p/2}$. For a parameter $r \le \ell_1$, we create 2^{-r}-fraction lists $\tilde{L}_i^{(2)} \subset L_{2i-1}^{(3)} \times L_{2i}^{(3)}$ for $1 \le i \le 4$, where each element is defined as follows:

$$
\begin{aligned}
\mathbf{z}_1^{(2)} &= \mathbf{z}_1^{(3)} + \mathbf{z}_2^{(3)} \text{ s.t. } \pi_r(\mathbf{H}_2\mathbf{z}_1^{(3)}) + \pi_r(\mathbf{H}_2\mathbf{z}_2^{(3)}) = \mathbf{t}_1, \\
\mathbf{z}_2^{(2)} &= \mathbf{z}_3^{(3)} + \mathbf{z}_4^{(3)} \text{ s.t. } \pi_r(\mathbf{H}_2\mathbf{z}_3^{(3)}) + \pi_r(\mathbf{H}_2\mathbf{z}_4^{(3)}) = \mathbf{t}_1, \\
\mathbf{z}_3^{(2)} &= \mathbf{z}_1^{(3)} + \mathbf{z}_2^{(3)} \text{ s.t. } \pi_r(\mathbf{H}_2\mathbf{z}_1^{(3)}) + \pi_r(\mathbf{H}_2\mathbf{z}_2^{(3)}) = \mathbf{t}_2, \\
\mathbf{z}_4^{(2)} &= \mathbf{z}_3^{(3)} + \mathbf{z}_4^{(3)} \text{ s.t. } \pi_r(\mathbf{H}_2\mathbf{z}_3^{(3)}) + \pi_r(\mathbf{H}_2\mathbf{z}_4^{(3)}) = \pi_r(\mathbf{s}_2) + \mathbf{t}_2,
\end{aligned}
$$

where $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{F}_2^r$. The time complexity for depth-2 lists is $\max(|D|^{1/2}, 2^{-r}|D|)$. The size of a depth-2 list is $|\tilde{L}^{(2)}| = \max(1, 2^{-r}|D|)$. For depth 1-lists, we obtain 2^{-r}-fraction lists $\tilde{L}_i^{(1)} \subset \tilde{L}_{2i-1}^{(2)} \times \tilde{L}_{2i}^{(2)}$ for $i = 1, 2$ with time $\max(|\tilde{L}^{(2)}|, 2^{r-\ell_1}|\tilde{L}^{(2)}|^2)$ and space $|\tilde{L}^{(1)}| = \max(1, 2^{r-\ell_1}|\tilde{L}^{(2)}|^2)$. Finally, we merge $\tilde{L}_1^{(1)}$ and $\tilde{L}_2^{(1)}$ in time $\max(|\tilde{L}^{(1)}|, 2^{\ell_1-\ell}|\tilde{L}^{(1)}|^2)$ and obtain a fraction of the permuted solution with probability $\rho_{\text{repr}} = \min(1, 2^{-\ell_1-2r}R)$, where, $R = \binom{2p}{p} \approx \binom{p}{p/2}^2$. When $2^{-\ell_1-2r}R < 1$, we need to iterate the Search component $2^{\ell_1+2r}R^{-1}$ times to yield one permuted solution expectedly under a good permutation. The asymptotic time complexity is

$$
q^{-1}\, {\rho_{\text{repr}}}^{-1} \max(|L^{(3)}|, |\tilde{L}^{(2)}|, |\tilde{L}^{(1)}|, 2^{\ell_1-\ell}|\tilde{L}^{(1)}|^2),
$$

where $q = \binom{k+\ell}{2p}\binom{n-k-\ell}{w-2p}\binom{n}{w}^{-1}$. The space complexity is $\max(|L^{(3)}|, |\tilde{L}^{(2)}|, |\tilde{L}^{(1)}|)$.

Numerical Optimization The asymptotic complexity of the time-memory trade-off MMT algorithm for full distance decoding is

$$T = 2^{0.111n} \text{ and } S = 2^{0.0376n},$$

at $\tilde{k} = 0.44$ and $\tilde{w} = 0.1273$, with optimal parameters of

$$\tilde{p} = 0.01073,\ \tilde{r} = 0,\ \tilde{\ell_1} = 0.03764,\ \tilde{\ell} = 0.07527.$$

We confirm that the Schroeppel–Shamir technique cannot reduce memory without sacrificing time complexity for the MMT algorithm. Nevertheless, it still results in almost the same space complexity as $2^{0.0375n}$, as derived in [20]. This implies that the time-memory trade-off term ρ_{repr} introduced in [20] plays a crucial role in reducing space complexity. Note that T is minimized when $\ell_1 = \log_2 \binom{(k+\ell)/2}{p/2}$, which leads to $S = |L^{(2)}| = |L^{(1)}|$.

Asymptotic Complexity of the Improved BJMM Algorithm The asymptotic complexity of the improved BJMM algorithm is the same as the depth-2 BJMM algorithm. This is because a specific weight distribution pair (i, j), which maximizes $|\mathcal{C}_{i,j}|\rho_{i,j}$ in Eq. (13), dominates over all weight distributions. The asymptotic complexity of the depth-2 BJMM algorithm is

$$T = 2^{0.105n} \text{ and } S = 2^{0.0659n},$$

at $\tilde{k} = 0.43$ and $\tilde{w} = 0.1273$, with optimal parameters

$$\tilde{p}' = 0.01076,\ \tilde{p} = 0.01812,\ \tilde{\ell_1} = 0.06588,\ \tilde{\ell} = 0.1318.$$

Note that T is minimized when $2^{\ell_1} = S = |L^{(2)}| = |L^{(1)}| = R$, where $|L^{(2)}| = \binom{(k+\ell)/2}{p'}$ and $R \approx \binom{2p}{p}\binom{k+\ell-2p}{2p'-p}$.

6 Cryptanalysis

In this section, we present security estimates of Classic McEliece, BIKE, and HQC for existing ISD algorithms. We use `CryptographicEstimators`[4], which is the latest cryptanalysis library developed by Esser et al. [19]. The SDP parameter sets we target are listed in Table 2.

6.1 Classic McEliece

First, we present the estimated bit time complexity and its corresponding space complexity for all parameter sets of Classic McEliece in Table 3.

In our cryptanalysis, we consider eight ISD algorithms including MMT-TMTO and BJMM-TMTO, which are time-memory trade-off variants of the MMT

[4] https://github.com/Crypto-TII/CryptographicEstimators

Table 2. Parameter sets for Classic McEliece, BIKE and HQC proposals.

Scheme	Category	n	k	w
Classic McEliece	1	3488	2720	64
	3	4608	3360	96
	5	6688	5024	128
	5	6960	5413	119
	5	8192	6528	128
BIKE (message)	1	24646	12323	134
	3	49318	24659	199
	5	81946	40973	264
BIKE (key)	1	24646	12323	142
	3	49318	24659	206
	5	81946	40973	274
HQC	1	35338	17669	132
	3	71702	35851	200
	5	115274	57637	262

and the BJMM algorithm [20]. BJMM$^+$ represents the improved BJMM algorithm. To derive the estimated complexity for Sieving ISD, we use open-source code provided by the authors[5]. The bold font indicates the minimal bit time complexity (T) or bit space complexity (M). Among these ISD algorithms, BJMM$^+$ achieves the smallest time complexity across all categories when assuming unlimited memory capacity and constant memory access cost. Additionally, BJMM$^+$ reduced bit security for Classic McEliece 3 by 11 bits from MMT-tmto, and 3 bits from BJMM-tmto.

In [18], the authors confirm that the assumption of the logarithmic memory access cost model aligns well with actual implementation. We also verify the validity of this assumption for our implementation. We also evaluate the security of NIST-PQC candidates under realistic memory constraints, considering both the logarithmic access model and a maximum memory capacity of 2^{43} or 2^{60} bits (equivalent to 1 terabyte and 155 petabytes), denoted as $\mathrm{BJMM}^+_{M\leq 43}$ and $\mathrm{BJMM}^+_{M\leq 60}$ in each table.

As a result, the security levels of Classic McEliece for Categories 1, 5a, 5b, and 5c have sufficiently large security margins from the security requirements (128 bits for Category 1, 192 bits for Category 3, and 256 bits for Category 5) when memory constraints are assumed. However, for Category 3, the security level remains below the desired security level for the BJMM$^+$ algorithm.

6.2 BIKE and HQC

Since both BIKE and HQC use a quasi cyclic code, it is known that the time complexities of several ISD algorithms can be decreased by leveraging the cyclic

[5] https://github.com/vunguyen95/Review-ISD-Sieving

Table 3. Estimated bit security and bit space complexity for Classic McEliece. Underlines indicate a deficiency in meeting the specified security requirements (128 bits for Category 1, 192 bits for Category 3, and 256 bits for Category 5).

Category	1 ($n = 3488$)		3 ($n = 4608$)		5a ($n = 6688$)		5b ($n = 6960$)		5c ($n = 8192$)	
	T	M	T	M	T	M	T	M	T	M
BJMM$^+$	**140**	98	**$\underline{179}$**	116	**$\underline{245}$**	146	**$\underline{245}$**	169	**275**	174
Prange	173	**22**	217	**23**	296	**24**	297	**24**	334	**24**
Dumer	151	58	193	60	268	89	268	90	303	109
MMT-tmto	148	59	190	70	261	90	261	91	294	102
BJMM-tmto	142	98	$\underline{182}$	122	$\underline{248}$	162	$\underline{248}$	160	277	189
May-Ozerov	141	87	$\underline{180}$	115	$\underline{246}$	165	$\underline{246}$	160	276	194
Both-May	142	88	$\underline{181}$	113	$\underline{248}$	143	$\underline{247}$	145	279	149
Sieving ISD	143	58	$\underline{184}$	65	257	91	257	92	291	95
BJMM$^+_{M \leq 43}$	147	43	$\underline{191}$	43	267	43	268	43	304	43
BJMM$^+_{M \leq 60}$	143	60	$\underline{186}$	55	261	58	261	59	297	60

nature of the code. We present the results of our security estimations for BIKE and HQC in Table 4 and 5, respectively.

For the key security of BIKE, the time complexities of all ISD algorithms are reduced by a factor of k without any additional effort. To attack the secret key of BIKE, we need to solve the quasi cyclic SDP, where the syndrome is the zero vector. This SDP contains k different solutions, which decrease the expected number of loops required for any ISD by a factor of k. For the bit security, we present the results with $\log_2 k$ subtracted from the estimations.

In the case of message security for BIKE and HQC, several ISD algorithms can reduce time complexity by implementing the Decoding-One-Out-of-Many (DOOM) strategy, as described in [35]. For Dumer, MMT-tmto, BJMM-tmto, and BJMM$^+$, we can reduce the time complexity by $\Omega(\sqrt{k})$, where $k = n/2$, by utilizing the asymmetrical base list construction technique, as shown in [18].

For May-Ozerov and Both-May, concrete algorithms for realizing DOOM speedups have not yet been developed. In this paper, a common assumption of $\sqrt{k}$ speedup is applied to them, as in [17]. For Sieving ISD, the authors claim k times speedups by leveraging the rotations of the syndrome while enlarging vectors in the search phase.

From Tables 4 and 5, both BIKE and HQC meet the desired level of bit security across all categories. The difference in time complexity between sieving ISD and other ISDs for quasi-cyclic codes stems mainly from discrepancies in the speedup gains of DOOM. To our knowledge, there is currently no practical evidence for the sieving ISD for quasi-cyclic SDP instances. Hence, in this paper, we also employ the improved BJMM algorithm for memory-constrained estimations. The verification of the sieving ISD for quasi-cyclic codes remains a future

Table 4. Estimated bit security and bit space complexity for BIKE.

	Category	1 ($n = 24646$)		3 ($n = 49318$)		5 ($n = 81946$)	
		T	M	T	M	T	M
key security	BJMM$^+$	146	54	210	59	277	62
	Prange	168	**28**	234	**30**	304	**32**
	Dumer	148	40	211	43	279	45
	MMT-tmto	148	38	211	40	279	41
	BJMM-tmto	147	55	211	57	278	61
	May-Ozerov	147	55	210	57	278	61
	Both-May	147	55	210	57	278	61
	Sieving ISD	**141**	46	**204**	50	**271**	53
	BJMM$^+_{M\le 43}$	146	42	211	43	280	41
	BJMM$^+_{M\le 60}$	146	44	210	47	277	50
message security	BJMM$^+$	145	46	210	59	275	63
	Prange	167	**28**	235	**30**	301	**32**
	Dumer	146	41	211	44	276	46
	MMT-tmto	146	38	211	40	276	41
	BJMM-tmto	146	38	211	40	276	61
	May-Ozerov	146	55	211	57	276	61
	Both-May	146	55	211	57	276	61
	Sieving ISD	**135**	46	**198**	50	**262**	53
	BJMM$^+_{M\le 43}$	145	42	212	40	278	41
	BJMM$^+_{M\le 60}$	145	43	210	46	275	48

challenge. When assuming a logarithmic memory access cost and constrained memory capacity, both BIKE and HQC still have sufficiently large security margins.

7 Experiments

This section describes details regarding our GPU implementation of the improved BJMM algorithm and the record computation for the McEliece-1409 instance.

7.1 A GPU Implementation of the BJMM Algorithm

We provide the improved BJMM algorithm as a Compute Unified Device Architecture (CUDA) implementation (`cuBJMM`). Our implementation is developed as an improved variant of the CUDA MMT implementation (`cuMMT` [30][6]). In

[6] The reference implementation for `cuMMT` is available at `https://www.jstage.jst.go.jp/article/transfun/E106.A/3/E106.A_2022CIP0023/_pdf/`

Table 5. Estimated bit security and bit space complexity for HQC.

Category	1 ($n = 35338$)		3 ($n = 71702$)		5 ($n = 115274$)	
	T	M	T	M	T	M
BJMM$^+$	145	48	213	52	275	55
Prange	166	**29**	236	**31**	300	**33**
Dumer	145	43	213	46	275	48
MMT-tmto	145	38	213	40	275	42
BJMM-tmto	145	38	213	40	275	42
May-Ozerov	146	39	214	42	276	44
Both-May	146	39	214	42	276	44
Sieving ISD	**141**	46	**204**	50	**271**	53
BJMM$^+_{M\leq 43}$	145	43	214	40	276	42
BJMM$^+_{M\leq 60}$	145	43	213	47	275	49

`cuBJMM`, only two lists $\bar{L}_1^{(2)}$ and $\bar{L}_1^{(1)}$, are constructed as simple one-dimensional integer arrays during the list construction phase. These arrays function as hash maps, with keys corresponding to $\pi_{\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(2)})$ or $\pi_{\ell-\ell_1}(\mathbf{H}_2\mathbf{z}_1^{(1)})$. When merging two lists, GPU threads virtually construct the other list by enumerating elements meeting the list constraint. We utilize asynchronous concurrent writing technique in list merging to enhance the effectiveness of parallel list merging. In addition, we deployed several improvements as listed below.

1. (CPU-GPU parallelism) The outer loop (permutation) in the improved BJMM algorithm is parallelized by multi-threading on CPU, i.e., we run N-BJMM procedures independently for N different permutations, where N is the number of CPU thread. Then, each BJMM routine borrows a memory segment on a GPU and executes parallel list construction.
2. (Fast Gaussian elimination) We use an optimized implementation of the *Method of Four Russians for Inversion* (M4RI) [1], improved by Esser, May and Zweydinger[7].
3. (Memory optimization) We transform the non-variable values into constants by using the `constexpr` feature in `C++` to reduce memory access costs.

With these improvements, `cuBJMM` achieved a 23.4× faster expected runtime compared to `cuMMT` for the McEliece-1409 instance.

7.2 Decoding McEliece-1409 Challenge

We estimated the bit complexity and optimal parameters for McEliece-1409 using `CryptographicEstimators`, under constraints of $M \leq 33$ (1 gigabyte) and the logarithmic access model in Table 6. It is observed that BJMM$^+$ is 3.3×

[7] Available at `https://github.com/FloydZ/cryptanalysislib`.

faster than BJMM-TMTO and still 1.9× faster than MAY-OZEROV, which yields the smallest time complexity for McEliece instances among the ISD algorithms in [17]. Note that there is no practical implementation for MAY-OZEROV and BOTH-MAY due to the low efficiency of the local sensitive hashing (LSH) technique. In our estimation the LSH cost is derived from the Indyk–Motwani nearest neighbor algorithm [23]. The space complexity of PRANGE corresponds to the size of parity-check matrix **H**.

Table 6. Estimated complexity and optimal parameters for McEliece-1409 with $M \leq 33$ (1 gigabyte) and the logarithmic memory access cost model.

Algorithm	T	M	p'	p	ℓ_1	ℓ	w_1	w_2	depth
BJMM$^+$	**70.1**	31.5	–	4	14	36	–	–	2
PRANGE	88.6	**18.6**	–	–	–	–	–	–	–
DUMER	72.4	28.8	–	2	–	19	–	–	1
MMT-TMTO	71.8	32.3	2	4	13	36	–	–	2
BJMM-TMTO	71.8	32.3	2	4	13	36	–	–	2
MAY-OZEROV	71.0	32.2	2	4	–	13	–	–	2
BOTH-MAY	71.1	32.2	2	4	–	13	0	0	2

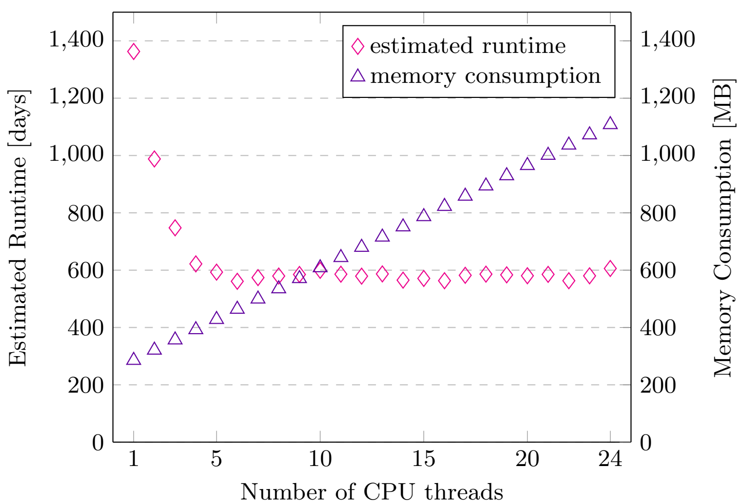

Fig. 3. Estimated runtime and memory consumption of `cuBJMM` for McEliece-1409 with varying the number of CPU threads on the desktop PC.

In practice, we use $p = 4, \ell_1 = 14, \ell = 35$ for our implementation to solve the McEliece-1409 instance, as it requires two large integer arrays of sizes 2^{ℓ_1} and

$2^{\ell-\ell_1}$. Figure 3 shows the expected runtime and the memory consumption. The estimated runtime is given by $T_{\text{loop}} q^{-1}$, where T_{loop} is the measured runtime for one iteration with `cuBJMM`. We parallelized 16 CPU threads in our experiments, requiring an expected 563 days and 822 megabytes ($2^{32.6}$ bits) on one desktop PC to solve McEliece-1409. The maximal number of GPU threads is set to $2^{-\ell_1}|L^{(2)}|^2 = 1,684,900$ per CPU thread, resulting in a total of $26,958,400$ GPU threads per PC.

With 10 desktop PCs (5 each equipped with an RTX 4080 GPU and an Intel Core i9-12900 CPU, and 5 with an RTX 3090 GPU and the same CPU), we achieve an expected runtime of 65.3 days for McEliece-1409. As a result, we solved the McEliece-1409 instance in 29.6 hours.

7.3 Validation of the Result

There may be concerns regarding the disparity between the expected runtime and the actual runtime for the McEliece-1409 record. To verify the correctness, we conducted a comprehensive experiment for the SDP with parameters $n = 808$, $k = 647$, and $w = 17$, whose bit complexity is $2^{49.5}$. The average number of iterations required to solve the instance is $q^{-1} = 2^{19.17}$. We solved the instance 10^3 times. Fig. 4 shows a histogram for the number of iterations.

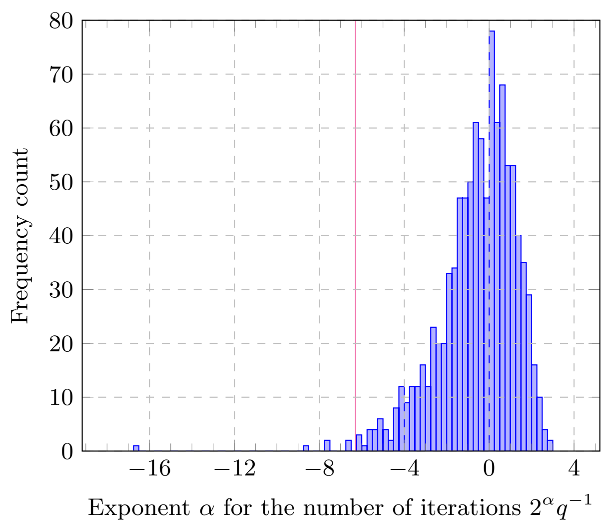

Fig. 4. Histogram of the exponent α for the number of iterations $2^{\alpha}q^{-1}$ required to solve the McEliece-808 instance for 10^3 trials. Here, $q^{-1} = 2^{19.17}$ represents the average number of iterations to solve the instance. The vertical red line represents $\alpha = -6.3$, which corresponds to our McEliece-1409 record.

The number of successful trials maximizes when the number of iterations approaches the average. No successes are observed for iterations exceeding 2^3 times the average. On the other hand, even for a number of iterations that is 2^8 times smaller than the average, there are still successes. For the McEliece-1409 record, $\alpha = -6.3$ is obtained, which is also depicted in the figure.

We also compute the probability of finding a solution within a specified runtime. The probability density function for the N-th iteration at which the algorithm terminates is given by $f(N) = q(1-q)^{N-1}$, which is the geometric distribution. Since the total time complexity of an ISD algorithm up to the N-th iteration is $N(T_{\text{ge}}+T_{\text{search}})$, $f(N)$ can be extended to a map between the runtime and the probability of success: $f(t) = q(1-q)^{t/T-1}$, where $T = T_{\text{ge}}+T_{\text{search}}$. The cumulative distribution function for $f(t)$ is $F(t) = 1-(1-q)^{t/T}$. Our interest is to determine two positions t_α and $t_{1-\alpha}$, where $F(t_\alpha) = \alpha$ and $F(t_{1-\alpha}) = 1-\alpha$, as illustrated in Figure 5.

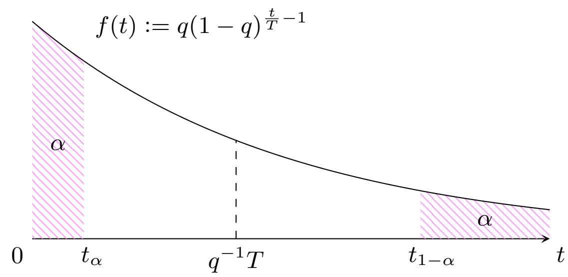

Fig. 5. This figure illustrates $t_\alpha \approx q^{-1}T\alpha$ and $t_{1-\alpha} \approx q^{-1}T\ln\alpha^{-1}$ w.r.t. a parameter $0 \le \alpha \le 1$. $f(t) := q(1-q)^{t/T-1}$ is a map from the runtime t to the success probability $f(t)$ at which an ISD algorithm terminates. We draw $f(t)$ with $q = 0.01$ for simplicity.

We can compute t_α by solving the following equation for t: $1-(1-q)^{t/T} = \alpha$, which gives

$$t_\alpha = \left(q^{-1}\alpha + O(q^{-1}\alpha^2)\right)T, \tag{18}$$

by series expansion. Assuming $q \ll 1$ and $\alpha \ll 1$, Eq. (18) is approximated by

$$t_\alpha \approx q^{-1}T\alpha. \tag{19}$$

Similarly, $t_{1-\alpha}$ can be determined by solving the following equation for t: $1-(1-q)^{t/T} = 1-\alpha$, which gives

$$t_{1-\alpha} = \left(q^{-1} - \frac{1}{2} + O(q)\right)T\ln\alpha^{-1}. \tag{20}$$

Assuming $q \ll 1$, then Eq. (20) is approximated by

$$t_{1-\alpha} \approx q^{-1}T\ln\alpha^{-1}. \tag{21}$$

It is noteworthy that Eq. (21) increases on a logarithmic scale with decreasing in α, whereas Eq. (19) linearly decreases as $q \ll 1$ is satisfied in ISD algorithms.

7.4 Comparison with Latest Implementations

We compare the performance of `cuBJMM` with other ISD implementations. Figure 6 shows estimated runtimes and bit complexities of `cuBJMM`, along with computation results. Below, we describe recent record computations for McEliece instances.

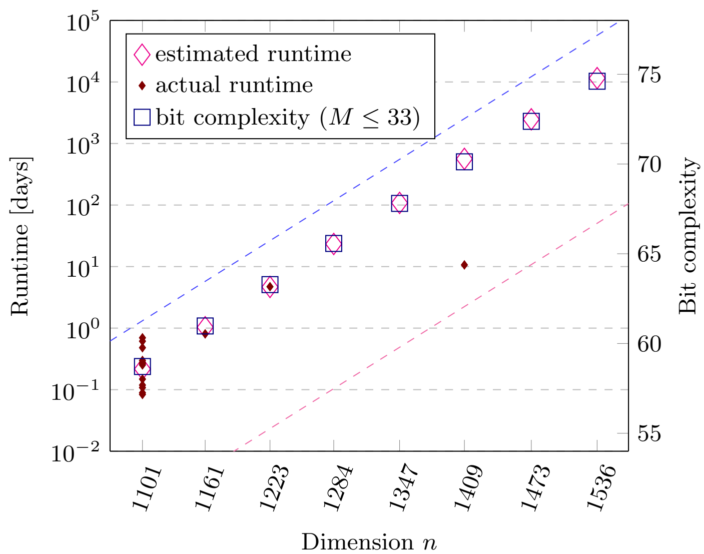

Fig. 6. Bit complexities and estimated running times to solve each McEliece challenge with one desktop PC equipped with an Intel Core i9-12900 CPU and an RTX 4080 GPU. Instances that were successfully solved by our implementation are marked with small squares. The red dashed line represents the runtime $q^{-1}T\alpha$ and the blue dashed line represents the runtime $q^{-1}T \ln \alpha^{-1}$ with a probability $\alpha = 2^{-8}$.

McEliece-1161 was solved in 15.66 days by Narisada, Fukushima, and Kiyomoto using a GPU implementation of Dumer's algorithm on an Intel Xeon E5-2686v4 server and an NVIDIA Tesla V100 [29].

Esser, May, and Zweydinger achieved the first records for McEliece-1223 and McEliece-1284 at 2.45 days and 31.43 days, respectively, using their fast implementation of the MMT/BJMM algorithm with 4 AMD EPYC 7742 CPUs [18]. Their implementation was later improved by introducing time-memory tradeoffs, achieving an expected runtime of 13.10 days for McEliece-1284 [20].

Recently, Bernstein, Lange, and Peters solved McEliece-1347 using software they developed on several clusters of computers [9]. According to the website[8], it is stated that the expected runtime of their implementation for McEliece-1284 with 4 AMD EPYC 7742 CPUs is 31.56 days.

[8] https://isd.mceliece.org/1347.html, published on February 26, 2023.

The expected runtimes of `cuBJMM` with one desktop PC under the memory constraints of $M \leq 33$ for McEliece-1284 and McEliece-1347 are 23.30 days and 108.39 days, respectively. For McEliece-1473 and McEliece-1536, expected runtime of our implementation with $M \leq 33$ are 2474 days and 11552 days, respectively.

8 Conclusion

In this paper, we propose an improved variant of the depth-2 BJMM algorithm. This algorithm offers the lowest bit security level for Classic McEliece among existing ISD algorithms. We also present the first publicly available GPU implementation of the improved BJMM algorithm. We solve McEliece-1409 for the first time in 30 hours using 10 desktop PCs. These results provide both theoretical and practical evidence for the reliability of code-based NIST-PQC round 4 candidates.

Future work should include concrete analysis for newer ISDs, such as the May–Ozerov, Both–May, and sieving ISD algorithms. It is important to verify the resilience of the remaining code-based NIST-PQC candidates against quantum ISD algorithms from both theoretical and practical perspectives.

Acknowledgments

This study was supported by JSPS KAKENHI JP22KJ0554 and the Joint Research Center for Advanced and Fundamental Mathematics for Industry, Institute of Mathematics for Industry, Kyushu University (2022a015).

References

1. Albrecht, M., Bard, G.: The M4RI Library. The M4RI Team (2023), https://bitbucket.org/malb/m4ri
2. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., Von Maurich, I., Misoczki, R., Niederhagen, R., et al.: Classic mceliece: conservative code-based cryptography (2022)
3. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., et al.: Bike: bit flipping key encapsulation (2022)
4. Aragon, N., Lavauzelle, J., Lequesne, M.: decodingchallenge.org (2019), http://decodingchallenge.org
5. Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., et al.: Supersingular isogeny key encapsulation. Submission to the NIST Post-Quantum Standardization project **152**, 154–155 (2017)
6. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: EUROCRYPT 2012. pp. 520–536 (2012)

7. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory **24**(3), 384–386 (1978)
8. Bernstein, D.J., Chou, T.: Cryptattacktester: high-assurance attack analysis. Cryptology ePrint Archive, Paper 2023/940 (2023), https://eprint.iacr.org/2023/940, https://eprint.iacr.org/2023/940
9. Bernstein, D.J., Lange, T., Peters, C.: Attacking and Defending the McEliece Cryptosystem. In: Buchmann, J., Ding, J. (eds.) Post-Quantum Cryptography. pp. 31–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
10. Both, L., May, A.: Decoding linear codes with high error rate and its impact for LPN security. In: International Conference on Post-Quantum Cryptography. pp. 25–46 (2018)
11. Canto Torres, R., Sendrier, N.: Analysis of Information Set Decoding for a Sublinear Error Weight. In: Takagi, T. (ed.) Post-Quantum Cryptography. pp. 144–161. Springer International Publishing, Cham (2016)
12. Carrier, K., Debris-Alazard, T., Meyer-Hilfiger, C., Tillich, J.P.: Statistical decoding 2.0: Reducing decoding to lpn. In: Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV. p. 477–507. Springer-Verlag, Berlin, Heidelberg (2023)
13. Castryck, W., Decru, T.: An Efficient Key Recovery Attack on SIDH. In: Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. p. 423–447. Springer-Verlag, Berlin, Heidelberg (2023)
14. Ducas, L., Esser, A., Etinski, S., Kirshanova, E.: Asymptotics and Improvements of Sieving for Codes. Cryptology ePrint Archive, Paper 2023/1577 (2023), https://eprint.iacr.org/2023/1577, https://eprint.iacr.org/2023/1577
15. Dumer, I.: On minimum distance decoding of linear codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory. pp. 50–52 (1991)
16. Esser, A.: Revisiting Nearest-Neighbor-Based Information Set Decoding. Cryptology ePrint Archive, Paper 2022/1328 (2023), https://eprint.iacr.org/2022/1328, https://eprint.iacr.org/2022/1328
17. Esser, A., Bellini, E.: Syndrome Decoding Estimator. In: Public-Key Cryptography – PKC 2022. pp. 112–141 (2022)
18. Esser, A., May, A., Zweydinger, F.: McEliece Needs a Break – Solving McEliece-1284 and Quasi-Cyclic-2918 with Modern ISD. In: Advances in Cryptology – EUROCRYPT 2022. pp. 433–457 (2022)
19. Esser, A., Verbel, J., Zweydinger, F., Bellini, E.: `CryptographicEstimators`: a Software Library for Cryptographic Hardness Estimation. Cryptology ePrint Archive, Paper 2023/589 (2023), https://eprint.iacr.org/2023/589, https://eprint.iacr.org/2023/589
20. Esser, A., Zweydinger, F.: New Time-Memory Trade-Offs for Subset Sum – Improving ISD in Theory and Practice. In: Advances in Cryptology – EUROCRYPT 2023. pp. 360–390 (2023)
21. Guo, Q., Johansson, T., Nguyen, V.: A New Sieving-Style Information-Set Decoding Algorithm. Cryptology ePrint Archive, Paper 2023/247 (2023), https://eprint.iacr.org/2023/247, https://eprint.iacr.org/2023/247
22. Hamdaoui, Y., Sendrier, N.: A Non Asymptotic Analysis of Information Set Decoding. IACR Cryptol. ePrint Arch. **2013**, 162 (2013), https://api.semanticscholar.org/CorpusID:17721683

23. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 604–613 (1998)
24. Kachigar, G., Tillich, J.P.: Quantum Information Set Decoding Algorithms. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography. pp. 69–89. Springer International Publishing, Cham (2017)
25. May, A., Meurer, A., Thomae, E.: Decoding Random Linear Codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: ASIACRYPT 2011. pp. 107–124 (2011)
26. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: EUROCRYPT 2015. pp. 203–228 (2015)
27. McEliece, R.J.: A Public-Key Cryptosystem Based On Algebraic Coding Theory. Deep Space Network Progress Report **44**, 114–116 (Jan 1978)
28. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Persichetti, E., Zémor, G., Bourges, I.: Hamming quasi-cyclic (hqc). NIST PQC Round **2**(4), 13 (2018)
29. Narisada, S., Fukushima, K., Kiyomoto, S.: Fast GPU Implementation of Dumer's Algorithm Solving the Syndrome Decoding Problem. In: IEEE ISPA 2021. pp. 971–977 (2021)
30. Narisada, S., Fukushima, K., Kiyomoto, S.: Multiparallel MMT: Faster ISD Algorithm Solving High-Dimensional Syndrome Decoding Problem. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E106.A**(3), 241–252 (2023). https://doi.org/10.1587/transfun.2022CIP0023
31. National Institute of Standards and Technology: PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates (2022), https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4
32. Peters, C.: Information-set decoding for linear codes over F q. In: International Workshop on Post-Quantum Cryptography. pp. 81–94 (2010)
33. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **8**(5), 5–9 (1962)
34. Schroeppel, R., Shamir, A.: A $T = O(2^{n/2})$, $S = O(2^{n/4})$ Algorithm for Certain NP-Complete Problems. SIAM Journal on Computing **10**(3), 456–464 (1981). https://doi.org/10.1137/0210033, https://doi.org/10.1137/0210033
35. Sendrier, N.: Decoding One Out of Many. In: Post-Quantum Cryptography. pp. 51–67 (2011)
36. Stern, J.: A method for finding codewords of small weight. In: Coding Theory and Applications. pp. 106–113 (1989)