Security analysis of the Classic McEliece, HQC and BIKE schemes in low memory

Yu Li\textsuperscript{a,b}, Li-Ping Wang\textsuperscript{a,b,}\textsuperscript{*}

\textsuperscript{a}State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
\textsuperscript{b}School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract

With the advancement of NIST PQC standardization, three of the four candidates in Round 4 are code-based schemes, namely Classic McEliece, HQC and BIKE. Currently, one of the most important tasks is to further analyze their security levels for the suggested parameter sets. At PKC 2022 Esser and Bellini restated the major information set decoding (ISD) algorithms by using nearest neighbor search and then applied these ISD algorithms to estimate the bit security of Classic McEliece, HQC and BIKE under the suggested parameter sets. However, all major ISD algorithms consume a large amount of memory, which in turn affects their time complexities. In this paper, we reestimate the bit-security levels of the parameter sets suggested by these three schemes in low memory by applying K-list sum algorithms to ISD algorithms. Compared with Esser-Bellini’s results, our results achieve the best gains for Classic McEliece, HQC, and BIKE, with reductions in bit-security levels of 11.09, 12.64, and 12.19 bits, respectively.

Keywords: Information Set Decoding, Syndrome Decoding, Code-based Cryptography, K-list.

1. Introduction

Coding-based cryptosystems have gained significant attention due to their ability to resist attacks from quantum computers and their success in the NIST PQC standardization process. Among the four candidates advancing...
to the 4th round of the process, three are code-based schemes, namely Classic McEliece, HQC, and BIKE. Currently, a crucial task is to further analyze their bit-security levels of these three schemes under the suggested parameter sets.

It is widely acknowledged that the security of most code-based cryptographic schemes relies primarily on the hardness of the syndrome decoding (SD) problem, which is known to be NP-hard. The best solvers of the SD problem are information set decoding (ISD) algorithms, which can be used to estimate the bit-security levels of the suggested parameter sets for these schemes. In 1962, Prange proposed the first ISD algorithm, which involved finding an error vector with a fixed weight by guessing an information set such that the coordinates of an error vector indexed by the set are error-free. Over the next 50 years, Prange’s algorithm was significantly improved by allowing a few entries of an error vector indexed by the information set to be error-affected and by introducing advanced techniques such as meet-in-the-middle, representation technology, nearest neighbor search, and others, in which the major ISD algorithms are Stern-1988, MMT-2011, BJMM-2012, MO-2015, and BM-2018. All these improved ISD algorithms have a common iterative structure, where each iteration attempts to retrieve an error vector with fixed weight distribution corresponding to a given syndrome, and the number of iterations required is determined by the average probability of success for one iteration.

Due to the introduction of additional parameters requiring optimization, ISD algorithms can be quite complex. In 2016, Torres and Sendrier developed a concise and asymptotic mathematical formula to determine the time complexity of the main ISD algorithms. Since then, this formula has been widely used by almost all code-based cryptographic schemes to roughly estimate the security levels of their parameter sets. Accurately estimating the bit security of current code-based schemes is an important task. Recently, Esser and Bellini introduced a new approach to restate the major ISD algorithms by utilizing nearest neighbor search. They then applied these redescribed ISD algorithms to estimate the concrete security levels of parameter sets suggested by Classic McEliece, HQC, and BIKE. Additionally, Esser, May, and Zwegdinger achieved two new computing records for McEliece-1223 and McEliece-1284 by implementing MMT’s or BJMM’s ISD algorithms. Based on these records, they estimated the hardness of breaking these three schemes under the round-3 suggested parameter sets. However, advanced ISD al-
algorithms require a large amount of memory, which in turn affects their time complexities. Recently, Esser and Zweydinger extrapolated the bit-security levels of the NIST round-4 parameter sets for Classic McEliece, HQC, and BIKE by proposing a time-memory trade-off for MMT algorithm [22]. Guo et al. also obtained the bit-security levels for these schemes by proposing a new ISD algorithm [23]. Therefore, our goal is to reestimate the bit-security levels of the parameter sets suggested by these three schemes in low memory by applying $K$-list sum algorithms proposed in [24] to ISD algorithms.

The solving algorithm for a $K$-list sum problem is essentially a general birthday algorithm first proposed by Wagner in 2002 [25], where $K$ is a power of 2. Howgrave-Graham and Joux then developed a class of exhaustive memory-efficient list sum algorithms for $K = 4$ in order to solve knapsacks [26], while Becker et al. focused on the scenario of $K = 16$ [27]. In 2012, Dinur et al. proposed an improvement to the general birthday algorithm by demonstrating ideal asymmetric partitions for "magic numbers" of $K$ such as 7 and 11, known as the $K$-dissection algorithm [28]. In 2019, Dinur unified all previous improvements and analyses of the general birthday algorithms into an algebraic framework [24]. The original general birthday algorithms are concerned with finding just one solution, but when multiple (exhaustive) solutions are needed in specific settings such as ISD algorithms, they are referred to as $K$-list sum algorithms by Dinur. $K$-list sum algorithms have shown good performance and have been applied to many search problems, including knapsacks [29] and the learning parity with noise (LPN) problem [30]. By using $K$-list sum algorithms in the ISD framework, Bricout et al. [31] reselected the parameter sets of the Wave scheme, a code-based digital signature scheme on a ternary field [32], to meet given security levels. Karpman-Lefevre [33] derived a time-memory trade-off for the parameter sets in the Wave scheme. Furthermore, Chailloux et al. used the ISD framework and Wagner’s general birthday algorithm to solve the SD problem endowed with the Lee metric [34]. Additionally, Wang and Liu improved the ISD algorithm under restricted memory by utilizing the $K$-dissection algorithms proposed in [28] [35]. Thus, the recent advancements in the $K$-dissection algorithms have motivated us to conduct this study.

Our Contributions. We firstly extend the 4-list sum algorithms proposed in [24] and then apply them to the ISD framework under three memory models, namely constant memory, logarithmic memory penalty, and cube-root memory penalty, to derive new bit-security levels for the Classic McEliece, HQC,
and BIKE schemes. Finally, we compare our results with the latest findings in [20]. Our best results for the bit-security levels of Classic McEliece, HQC, and BIKE schemes, under a $2^{60}$-bit limitation for memory, are 11.09, 12.64, and 12.19 bits lower than those in [20], respectively. Furthermore, our algorithms reveal that, for the case of constant memory, the parameter sets for $n = 4608$, $n = 6688$, and $n = 6960$ of the Classic McEliece scheme do not achieve the claimed security levels, whereas the results in [20] show that only the parameter set for $n = 4608$ fails to achieve the claimed security level.

The paper is structured as follows: Section 2 provides definitions, notations, and lemmas that will be used throughout the paper. In Section 3, we analyze the memory consumption and time complexity of different versions of 4-list sum algorithms applied to the ISD framework. In Section 4, we calculate the new bit-security levels for the Classic McEliece, HQC, and BIKE schemes and compare them to the latest results in [22]. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Notations

A finite field with two elements is denoted as $\mathbb{F}_2$. If not specified explicitly, the vectors and matrices are represented by the bold lowercase and uppercase letters, respectively. The concatenation of two row vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is denoted as $(\mathbf{v}_1|\mathbf{v}_2)$. For simplicity, let the set \{1, 2, ..., $n$\} be $[n]$. $\mathbf{I}_k$ is used to denote the identity matrix with size $k$. The Hamming weight of a vector $\mathbf{v}$ is written as $\text{wt}(\mathbf{v})$. Given a length-$n$ vector $\mathbf{v}$ and a matrix $\mathbf{M}$ with $n$ columns, $\mathbf{M}_S$ denotes the matrix composed of the columns of $\mathbf{M}$ indexed by $S$, where $S$ is a subset of $[n]$. If $\mathcal{C}$ is a set consisting of vectors of length $n$, then define $\mathcal{C}_S = \{\mathbf{v}_S : \mathbf{v} \in \mathcal{C}\}$. The size of the set $S \subseteq [n]$ is denoted as $|S|$, and so we give a mapping $\delta_{S,[n]} : \mathbb{F}_2^{|S|} \rightarrow \mathbb{F}_2^n$ by $\delta_{S,[n]}(\mathbf{u}) = \mathbf{v}$ such that $\mathbf{v}_S = \mathbf{u}$ and $\mathbf{v}_{[n]\setminus S} = \mathbf{0}$, where $\mathbf{u} \in \mathbb{F}_2^{|S|}$, $\mathbf{v} \in \mathbb{F}_2^n$.

2.2. Syndrome decoding problems

In this subsection, we introduce the syndrome decoding problem, ISD framework and some needed definitions of in coding theory.

Definition 1. A subspace of $\mathbb{F}_2^n$ with dimensional $k$ is called as a binary linear code $\mathcal{C}$ with length $n$ and dimension $k$, which is generally denote as an $[n,k]$ code.
Definition 2. If the rows of a matrix $G \in F_2^{k \times n}$ form a basis of an $[n, k]$ code $C$, then the matrix $G$ is called as a generator matrix of code $C$. Meanwhile, a matrix $H \in F_2^{(n-k) \times n}$ is a parity-check matrix of $C$ if its kernel corresponds to an $[n, k]$ code $C$.

Definition 3. A syndrome $s$ of a random vector $v \in F_2^n$ is obtained as $s = vH^\top$, where $H$ is a parity-check matrix of an $[n, k]$ code, and one has $v \in C$ if and only if $vH^\top = 0$.

Problem 1 ($(n, k, t)$-SD problem). Input a parity-check matrix $H \in F_2^{(n-k) \times n}$ of an $[n, k]$ linear code $C$, a syndrome $s \in F_2^{n-k}$ and a given positive integer $t$, output an error vector $e \in F_2^n$ such that $He^\top = s^\top$ and $wt_H(e) = t$.

The SD problem plays an important role in coding theory, and also many variants of this problem were proposed in some cryptography scenarios. In order to describe a general framework of ISD algorithms more clearly, we introduce another problem, i.e. the multiple-solution syndrome decoding (MSSD) problem, which is equivalent to the subset sum problem in [31] and the checkable multiple syndrome decoding problem in [34].

Problem 2 ($(n, k, t, m)$-MSSD problem). Input a parity-check matrix $H \in F_2^{(n-k) \times n}$ of an $[n, k]$ code $C$, a syndrome $s \in F_2^{n-k}$, and a given positive integer $t$, output a set $\{e \in F_2^n : He^\top = s^\top, wt_H(e) = t\}$ with size $2^m$.

2.3. The general framework of ISD algorithms

Before stating the general framework of ISD algorithms, we firstly give the definition of the information set of a binary linear code as follows.

Definition 4. An information set of an $[n, k]$ code $C$ is a size-$k$ subset $I \subseteq [n]$ is called if $|C_I| = |C|$.

All of the advanced ISD algorithms of solving Problem 1, such as MMT’s and BJMM’s, intrinsically consist of the following framework with four steps:

1) **Permutation step.** Choose a random permutation matrix $P \in F_2^{n \times n}$ such that $(HP)_{[n]\setminus[k+\ell]}$ is partially reversible.

2) **Partial Gaussian elimination step.** Choose an invertible transformation matrix $U \in F_2^{(n-k) \times (n-k)}$ satisfying

$$UHP = \begin{pmatrix} \hat{H} & 0 \\ \hat{H} & I_{n-k-\ell} \end{pmatrix}, \quad Us^\top = \begin{pmatrix} \hat{s} \\ \check{s} \end{pmatrix},$$
where $\tilde{H} \in \mathbb{F}_2^{(k+\ell) \times (k+\ell)}$, $\tilde{H} \in \mathbb{F}_2^{(n-k-\ell) \times (k+\ell)}$, $\hat{s} \in \mathbb{F}_2^k$, and $\tilde{s} \in \mathbb{F}_2^{n-k-\ell}$. Meanwhile, the error vector $e$ is split into two vectors $\hat{e} \in \mathbb{F}_2^{k+\ell}$ and $\tilde{e} \in \mathbb{F}_2^{n-k-\ell}$, i.e., $e\tilde{P} = (e^\top | \tilde{e}^\top)$. Thus, we have

$$\tilde{H} \hat{e} = \hat{s}, \quad (1)$$
$$\tilde{H} \hat{e} + \tilde{e} = \tilde{s}. \quad (2)$$

3) **MSSD step.** Solve a $(k+\ell, \ell, p, m)$-MSSD problem, i.e., find $2^m$ error vectors $\hat{e}$ with $wt(\hat{e}) = p$ satisfying Equation (1).

4) **Test step.** For all of the solutions of MSSD step, compute $\tilde{e} = \tilde{H} \hat{e} + \tilde{s}$ according to Equation (2). If there exists $wt(\tilde{e}) = t - p$, then output $e = (e^\top | \tilde{e}^\top)\tilde{P}^{-1}$. Else, go back to step 1).

The above framework is an iterative structure where each iteration is to size $k$ an error vector with a fixed weight distribution and the number of the iteration depends on the success probability of Test step. According to the result in [34], the probability of success in step 4) is

$$P_{\text{succ}}^{-1} = \max \left\{ 1, \frac{\min \left\{ \binom{n}{t} \cdot 2^{-\ell}, 2^{n-k-\ell} \right\}}{2^m \cdot \binom{n-k-\ell}{t-p}} \right\}.$$  

Thus, the running time and memory of the above framework are derived in the following lemma.

**Lemma 1 ([34]).** For positive integer parameters $n, k, t, \ell, p, m$, if the time complexities of the partial Gaussian elimination and solving a $(k+\ell, \ell, p, m)$-MSSD problem are $T_G$ and $T_{MSSD}$, respectively, then an $(n, k, t)$-SD problem is solved in time complexity

$$T_{\text{ISD}} = O \left( P_{\text{succ}}^{-1} \cdot \max \{ T_G, T_{MSSD} \} \right),$$

where $P_{\text{succ}}^{-1}$ is defined as above.

2.4. **K-list sum algorithms**

Firstly we introduce the $K$-list sum problem as follows and then describe the solving algorithms.
Problem 3 (K-list sum problem). Input K sorted lists $L_1, \ldots, L_K$, where each set $L_i$ contains $2^m$ uniform random vectors of length $n$, output a set \$\{(y_1, \ldots, y_K) \in L_1 \times \cdots \times L_K : y_1 + \cdots + y_K = 0\}\$.

In this paper, we consider a solving algorithm of a 4-list sum problem over $\mathbb{F}_2$. In [24], author proposed a 4-list sum algorithm, where the underlying lists has the equal sizes, i.e., $|L_1| = |L_2| = |L_3| = |L_4|$. Here, we general the algorithm by setting

\[ |L_1| = |L_3| = 2^{m_1} \geq |L_2| = |L_4| = 2^{m_2}. \]

In the following, we describe the 4-list sum algorithm in Algorithm 1 in pseudo-code form.

**Algorithm 1** (4-list sum algorithm)

**Input:** three integers $m_1, m_2$ and $0 \leq v \leq m_2$ and 4 sorted lists $L_1, L_2, L_3, L_4$ with $|L_1| = |L_3| = 2^{m_1} \geq |L_2| = |L_4| = 2^{m_2}$, where each set $L_i$ is composed of uniformly random vectors of length $n = m_1 + m_2 + v$.

**Output:** a size-2$^{m_1}$ set $L_{1234} = \{(x_1, \ldots, x_4) \in L_1 \times \cdots \times L_4 : x_1 + \cdots + x_4 = 0\}$.

1: Initialize $L_{12}, L_{34}, L_{1234} \leftarrow \emptyset$. 
2: for all $(0|x) \in 0^{m_2-v} \times \mathbb{F}_2^v$ do 
3: \hspace{0.5cm} for all $x_1 \in L_1$ do 
4: \hspace{1cm} $L_{12} \leftarrow L_{12} \cup \{x_1 + x_2 : (x_1 + x_2)_[m_2] = (0|x), x_2 \in L_2\}$ 
5: \hspace{1cm} end for 
6: \hspace{0.5cm} for all $x_3 \in L_3$ do 
7: \hspace{1cm} $L_{34} \leftarrow L_{34} \cup \{x_3 + x_4 : (x_3 + x_4)_[m_2] = (0|x), x_4 \in L_4\}$ 
8: \hspace{1cm} end for 
9: \hspace{0.5cm} for all $x_{12} \in L_{12}$ do 
10: \hspace{1cm} $L_{1234} \leftarrow L_{1234} \cup \{x_{12} + x_{34} : (x_{12} + x_{34})_{[n]\backslash [m_2]} = 0, x_{34} \in L_{34}\}$ 
11: \hspace{0.5cm} end for 
12: end for

For a 4-list sum algorithm, the four given lists are randomly combined in pairs to generate two intermediate sets such that $m_2 - v$ entries are zero and the other $v$ entries are randomly selected from $\mathbb{F}_2$ for each element of the intermediate sets, and then the final solution set is generated by finding collisions between the elements of the two intermediate sets on the remaining entries and traversing all choices on the $v$ entries. Note that the intermediate
and final sets constructed by the meet-in-the-middle technology have the equal size $2^{m_1}$. Next, we illustrate this process in the following Figure.

Then, the running time and memory cost of Algorithm 1 are given as follows.

**Lemma 2.** The running time and memory of Algorithm 1 are $T_4$ and $M_4$, respectively, i.e.,

$$T_4 = O(2^{m_1+v}), \quad M_4 = O(2^{m_1}).$$

**Proof.** Since the elements in the sets $L_1$ and $L_2$ are randomly uniform and $|L_1| = 2^{m_1} \geq |L_2| = 2^{m_2}$, then we have

$$|L_{12}| = \frac{|L_1| \cdot |L_2|}{2^{m_2}} = \frac{2^{m_1} \cdot 2^{m_2}}{2^{m_2}} = 2^{m_1}.$$

Thus, Lines 3-5 in Algorithm 1 use the meet-in-the-middle to construct the set $L_{12}$ from $L_1$ and $L_2$ with the time complexity

$$T_{12} = \max\{|L_1|, |L_2|, |L_{12}|\} = 2^{m_1}.$$

Similarly, the time complexities of constructing the sets $L_{34}$ and $L_{1234}$ are

$$T_{34} = T_{1234} = 2^{m_1}$$

where the sets $L_{34}$ and $L_{1234}$ have the equal size $2^{m_1}$. 

Figure 1: 4-list sum algorithm with memory $2^{m_1}$ and length $n$ with $m_1 \geq m_2$. 

![Diagram of 4-list sum algorithm with memory $2^{m_1}$ and length $n$ with $m_1 \geq m_2$.](image)
The above process was repeated $2^v$ times and so the total time complexity is

$$T_4 = 2^v \cdot \max\{T_{12}, T_{34}, T_{1234}\} = 2^{m_1 + v},$$

and the total memory consumption is

$$M_4 = \max\{|L_1|, |L_2|, |L_3|, |L_4|, |L_{12}|, |L_{1234}|\} = 2^{m_1},$$

3. Our ISD algorithms

In this section, we firstly sketch a general ISD framework by 4-list sum algorithm, and then analyze the time complexities of the ISD algorithms by applying four different versions of the 4-list sum algorithm, respectively.

According to Section 2.3, almost all ISD algorithms can be represented as a framework with 4 steps, i.e., 1) Permutation step, 2) Partial Gaussian elimination step, 3) MSSD step and 4) Test step. Here, we adopt the 4-list sum algorithm in the step 3).

3.1. Applying 4-list sum algorithm to the ISD framework

For an $(n, k, t)$-SD problem, after the steps 1) and 2) of the general ISD algorithm, two equations are yielded as follows:

$$\hat{H}e = \hat{s}, \quad (3)$$
$$\tilde{H}e + \tilde{e} = \tilde{s}. \quad (4)$$

where $\ell < n - k$ is a positive integer, $\hat{H} \in \mathbb{F}_2^{\ell \times (k+\ell)}$, $\tilde{H} \in \mathbb{F}_2^{(n-k-\ell) \times (k+\ell)}$, $\hat{s} \in \mathbb{F}_2^\ell$, $\tilde{s} \in \mathbb{F}_2^{n-k-\ell}$, $\hat{e} \in \mathbb{F}_2^{k+\ell}$, $\tilde{e} \in \mathbb{F}_2^{n-k-\ell}$ and $wt(\hat{e}) = p$. Next, we use the 4-list sum algorithm to find $\hat{e}$ satisfying Equation (3).

Firstly, we need to divide $\hat{H}$ and $\hat{e}$ into four disjoint parts, respectively, i.e.,

$$\hat{H}e = \begin{pmatrix} H_1 & H_2 & H_3 & H_4 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = H_1e_1 + H_2e_2 + H_3e_3 + H_4e_4 = \hat{s},$$
and then construct the four sets as follows:

$$L_i = \left\{ (e_i, H_i e_i) : H_i \in \mathbb{F}_2^{\ell \times n}, e_i \in \mathbb{F}_{2^n}, wt(e_i) = p_i \right\}$$

for $i = 1, 2, 3$ and

$$L_4 = \left\{ (e_4, H_4 e_4 + \hat{s}) : H_i \in \mathbb{F}_2^{\ell \times n_4}, e_4 \in \mathbb{F}_{2^{n_4}}, wt(e_4) = p_4 \right\},$$

where $n_1 + n_2 + n_3 + n_4 = k + \ell$, $p_1 + p_2 + p_3 + p_4 = p$ and the sizes of the sets are

$$|L_i| = \binom{n_i}{p_i} = 2^{m_i}, \quad i = 1, 2, 3, 4.$$

For the second components of the sets $L_1, L_2, L_3, L_4$, whenever a 4-tuple $(H_1 e_1, H_2 e_2, H_3 e_3, H_4 e_4)$ satisfying the condition $H_1 e_1 + H_2 e_2 + H_3 e_3 + H_4 e_4 + \hat{s} = 0$ is found, record and store the corresponding 4-tuple $(e_1, e_2, e_3, e_4)$. If we set

$$|L_1| = |L_3| = 2^{m_1} \geq |L_2| = |L_4| = 2^{m_2},$$

by using the 4-list sum algorithm of Algorithm 1, we obtain $2^{m_1}$ solutions $\hat{e} = (e_1^T | e_2^T | e_3^T | e_4^T)$ satisfying Equation 2.

Next, run the step 4) of the general ISD framework to obtain the final solution to the $(n, k, t)$-SD problem. The above process is described in Algorithm 2 in pseudo-code form.
Algorithm 2 (ISD by 4-list sum algorithm)

Input: a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$, a syndrome $s \in \mathbb{F}_2^{n-k}$ and the parameters $t, m_1, \ell, v, p_1, n_i$ for $i = 1, 2, 3, 4$, which satisfy $m_1 = \binom{n_1}{p_1} = \binom{n_2}{p_2}$, $m_2 = \binom{n_3}{p_3}$, and $\ell = m_1 + m_2 + v$.

Output: an error vector $e \in \mathbb{F}_2^{n-k}$ satisfying $He^\top = s^\top$ and $wt(e) = t$.

1: Choose a set $I \in [n]$ of size $k + \ell$ containing an information set.
2: Partition $I$ into four disjoint sets $I_1, I_2, I_3, I_4$ with $|I_i| = n_i$ for $i = 1, 2, 3, 4$.
3: Choose an binary invertible matrix $U \in \mathbb{F}_2^{(n-k)\times(n-k)}$, which satisfies

$$UH_{[n]\setminus I} = \begin{pmatrix} 0 \\ I_{n-K-\ell} \end{pmatrix}, Us^\top = \begin{pmatrix} \tilde{s} \\ \hat{s} \end{pmatrix},$$

where $\tilde{H} \in \mathbb{F}_2^{(k+\ell)\times (k+\ell)}$, $\hat{H} \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}$, $\hat{s} \in \mathbb{F}_2^k$, $\tilde{s} \in \mathbb{F}_2^{n-k}$.

4: Let $H_i = (\hat{H})_{I_i}$ for $i = 1, 2, 3, 4$. Build four sets

$$L_i = \left\{ (e_i, H_i e_i) : H_i \in \mathbb{F}_2^{(n-k)\times n_i}, e_i \in \mathbb{F}_2^{n_i}, wt(e_i) = p_i \right\}, \quad i = 1, 2, 3, 4,$$

$$L_4 = \left\{ (e_4, H_4 e_4 + \tilde{s}) : H_4 \in \mathbb{F}_2^{(n-k)\times n_4}, e_4 \in \mathbb{F}_2^{n_4}, wt(e_1) = p_4 \right\}.$$

5: Initialize $L_{12}, L_{34}, L_{1234} \leftarrow \emptyset$.
6: for all $(0|y) \in 0^{m_2-v} \times \mathbb{F}_2^v$ do
7:  for all $(e_1, y_1) \in L_1$ do
8:   $L_{12} \leftarrow L_{12} \cup \{ (e_{12}, H_{12} e_{12}) : I_{12} = I_1 \cup I_2, \ e_{12} = \delta_{I_1,I_{12}}(e_1) + \delta_{I_2,I_{12}}(e_2), \ H_{12} = \hat{H}_{I_{12}}, \ (y_1 + y_2)[m_2] = (0|y), \ (e_2, y_2) \in L_2 \}.$
9:  end for
10: for all $(e_3, y_3) \in L_3$ do
11:   $L_{34} \leftarrow L_{34} \cup \{ (e_{34}, H_{34} e_{34} + \tilde{s}) : I_{34} = I_3 \cup I_4, \ H_{34} = \hat{H}_{I_{34}}, \ e_{34} = \delta_{I_3,I_{34}}(e_1) + \delta_{I_4,I_{34}}(e_4), \ (y_3 + y_4)[m_2] = (0|y), \ (e_4, y_4) \in L_4 \}.$
12: end for
13: for all $(e_{12}, y_{12}) \in L_{12}$ do
14:   $L_{1234} \leftarrow L_{1234} \cup \{ (\hat{e} = \delta_{I_{12},I}(e_{12}) + \delta_{I_{34},I}(e_{34}) : (y_{12} + y_{34})[m_2] = 0, \ (e_{34}, y_{34}) \in L_{34} \}.$
15: end for
16: end for
17: for all $\hat{e} \in L_{1234}$ do
18:  if $wt(\tilde{s} - \hat{H}\hat{e}) = t - (p_1 + p_2 + p_3 + p_4)$ then
19:    Output: $e = \delta_{I,[n]}(\hat{e}) + \delta_{[n]\setminus I,[n]}(\tilde{s} - \hat{H}\hat{e})$.
20:  else
21:    Return to Step 1.
22: end if
23: end for
3.2. Analyzing the complexities

In this section, we firstly derive the details of the time and memory complexities for Algorithm 2, and then give four instantiations of the parameters.

**Theorem 1.** Let $|L_1| = |L_3| = 2^{m_1} \geq |L_2| = |L_4| = 2^{m_2}$ in Algorithm 2. The ISD by a 4-list sum algorithm solves an $(n, k, t)$-SD problem with the time complexity $T_{ISD} = \ldots$ and memory consumption $M_{ISD}$, where

$$T_{ISD} = O\left( \max\{n(n - k)^2, 2^{m_1+v}\} \cdot \max\left\{ 1, \frac{\max\{1, \min\left\{\binom{n}{t}, 2^{-\ell}, 2^{n-k-\ell}\right\}\}}{\binom{n-k-\ell}{t-p}} \cdot 2^{m_1} \right\} \right)$$

$$M_{ISD} = O\left( \max\{n(n - k), 2^{m_1}\} \right),$$

with $0 \leq p \leq \min\{(k + \ell), t\}$, $0 \leq v \leq m_2$, $\ell = m_1 + m_2 + v$.

**Proof.** Lines 1-3 in Algorithm 2 represent the partial Gaussian elimination step in the general ISD algorithm, which has the time complexity

$$T_G = (n - k)^2(n + 1).$$

Note that the sets $L_1, L_2, L_3, L_4$ have the sizes

$$|L_1| = |L_3| = 2^{m_1} \geq |L_2| = |L_4| = 2^{m_2},$$

and then lines 4-16 solve a $(k + \ell, \ell, p, m_1)$-MSSD problem by using the 4-list sum algorithm described in Algorithm 1. According to Lemma 2, the time complexity and memory are

$$T_{MSSD} = 2^{m_1+v}, M_{MSSD} = 2^{m_1}.$$
Version 3 (V3): \( \left\lceil \frac{(k+\ell)/5}{p/5} \right\rceil = 2^m_1, \left\lceil \frac{(k+\ell)/10}{p/10} \right\rceil = 2^m_2 \),

Version 4 (V4): \( \left\lceil \frac{(k+\ell)/7}{p/7} \right\rceil = 2^m_1, \left\lceil \frac{(k+\ell)/14}{p/14} \right\rceil = 2^m_2 \),

where \( \ell \) needs to satisfy \( \ell = m_1 + m_2 + v \) according to Theorem 1.

4. Numerical Results

4.1. Bit-security levels

In this section, we used four instantiations of Algorithm 2 to reestimate the bit securities of the parameters suggested by the Classic McEliece, HQC and BIKE schemes.

Since all advanced ISD algorithms consume an exponential amount of memory while improving the time complexity, and the large memories actually affect the time complexity during the implementation of ISD algorithms. Therefore, both McEliece team [36] and [21] suggested that one should penalize the time complexity with memory appropriately when considering the specific security level of the scheme, i.e., multiplying the running time by some type of penalty factor. In order to compare with the experimental results in [20], we use three memory models, i.e., constant memory, logarithmic memory penalty and cube-root memory penalty, and so the corresponding results are given as \( T_{ISD}, T_{ISD} \cdot \log M_{ISD} \) and \( T_{ISD} \cdot \sqrt[3]{M_{ISD}} \) in the following tables.

In addition, for the HQC and BIKE schemes with the cyclic structures, [20] and [22] used the Decoding-One-Out-of-Many (DOOM) technique to obtain a speedup by a factor of roughly \( \sqrt{k} \) in the running time of the ISD algorithm. In this paper, we still adopt the DOOM technique to obtain bit-security levels for HQC and BIKE.

In the above tables, “V1”, “V2”, “V3”, and “V4” denote four instantiations of 4-list sum algorithms in Section 3.2, respectively, and “M < 60” indicates that the final time complexities of the ISD algorithms are computed with the constant memory no more than 60 bits. Meanwhile, “logarithmic” and “cube-root” represent the time complexities of ISD algorithms under logarithmic memory penalty and cube-root memory penalty, respectively.

In order to compare with the results in [20], the values in parentheses refer to the gain of the bit-security levels obtained by our algorithms relative to that in [20], i.e., \( T_{20} - T_{ours} \).
<table>
<thead>
<tr>
<th>Table 1: Bit-security levels of Classic McEliece.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category 1</strong></td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td><strong>Constant:</strong></td>
</tr>
<tr>
<td><strong>M ≤ 60</strong></td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>23</td>
</tr>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td><strong>Logarithmic:</strong></td>
</tr>
<tr>
<td><strong>M ≤ 60</strong></td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td><strong>Cube-root:</strong></td>
</tr>
<tr>
<td><strong>M ≤ 60</strong></td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td>Category</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td><strong>constant:</strong> $M \leq 61$</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td><strong>logarithmic:</strong></td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td><strong>cube-root:</strong></td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-message</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>BIKE-key</td>
</tr>
<tr>
<td>HQC</td>
</tr>
<tr>
<td>HQC</td>
</tr>
</tbody>
</table>
Table 3: The parameters in Table 1

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 3</th>
<th>Category 5a</th>
<th>Category 5b</th>
<th>Category 5c</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
</tr>
<tr>
<td>constant:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>(57, 12, 5)</td>
<td>(60, 12, 6)</td>
<td>(63, 12, 5)</td>
<td>(76, 16, 0)</td>
</tr>
<tr>
<td></td>
<td>(78, 16, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V2</td>
<td>(52, 12, 0)</td>
<td>(55, 12, 0)</td>
<td>(58, 12, 0)</td>
<td>(58, 12, 0)</td>
</tr>
<tr>
<td></td>
<td>(86, 16, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td>(45, 10, 0)</td>
<td>(47, 10, 0)</td>
<td>(49, 10, 0)</td>
<td>(50, 10, 0)</td>
</tr>
<tr>
<td></td>
<td>(51, 10, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V4</td>
<td>(60, 14, 0)</td>
<td>(62, 14, 0)</td>
<td>(67, 14, 0)</td>
<td>(67, 14, 0)</td>
</tr>
<tr>
<td></td>
<td>(69, 14, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logarithmic:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>(57, 12, 5)</td>
<td>(60, 12, 6)</td>
<td>(63, 12, 5)</td>
<td>(76, 16, 0)</td>
</tr>
<tr>
<td></td>
<td>(78, 16, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V2</td>
<td>(52, 12, 0)</td>
<td>(55, 12, 0)</td>
<td>(58, 12, 0)</td>
<td>(58, 12, 0)</td>
</tr>
<tr>
<td></td>
<td>(86, 16, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td>(45, 10, 0)</td>
<td>(47, 10, 0)</td>
<td>(49, 10, 0)</td>
<td>(50, 10, 0)</td>
</tr>
<tr>
<td></td>
<td>(51, 10, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V4</td>
<td>(60, 14, 0)</td>
<td>(62, 14, 0)</td>
<td>(67, 14, 0)</td>
<td>(67, 14, 0)</td>
</tr>
<tr>
<td></td>
<td>(69, 14, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cube-root:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>(49, 8, 13)</td>
<td>(52, 8, 14)</td>
<td>(63, 12, 5)</td>
<td>(63, 12, 5)</td>
</tr>
<tr>
<td></td>
<td>(65, 12, 5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V2</td>
<td>(52, 12, 0)</td>
<td>(55, 12, 0)</td>
<td>(58, 12, 0)</td>
<td>(58, 12, 0)</td>
</tr>
<tr>
<td></td>
<td>(60, 12, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V3</td>
<td>(45, 10, 0)</td>
<td>(47, 10, 0)</td>
<td>(49, 10, 0)</td>
<td>(50, 10, 0)</td>
</tr>
<tr>
<td></td>
<td>(51, 10, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V4</td>
<td>(60, 14, 0)</td>
<td>(62, 14, 0)</td>
<td>(67, 14, 0)</td>
<td>(67, 14, 0)</td>
</tr>
<tr>
<td></td>
<td>(69, 14, 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: The parameters in Table 2

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 3</th>
<th>Category 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
<td>((\ell, p, v))</td>
</tr>
<tr>
<td>constant:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIKE-message V1</td>
<td>(68, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>BIKE-key V1</td>
<td>(65, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>HQC V1</td>
<td>(68, 8, 20)</td>
<td>(73, 8, 21)</td>
</tr>
<tr>
<td>logarithmic:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIKE-message V1</td>
<td>(68, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>BIKE-key V1</td>
<td>(65, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>HQC V1</td>
<td>(68, 8, 20)</td>
<td>(73, 8, 21)</td>
</tr>
<tr>
<td>cube-root:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIKE-message V1</td>
<td>(68, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>BIKE-key V1</td>
<td>(65, 8, 19)</td>
<td>(70, 8, 20)</td>
</tr>
<tr>
<td>HQC V1</td>
<td>(68, 8, 20)</td>
<td>(73, 8, 21)</td>
</tr>
</tbody>
</table>

16
Table 5: Security parameters of Classic McEliece, HQC and BIKE.

<table>
<thead>
<tr>
<th>Category</th>
<th>1</th>
<th>3</th>
<th>5(a)</th>
<th>5b</th>
<th>5c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic McEliece</td>
<td>n</td>
<td>3488</td>
<td>4608</td>
<td>6688</td>
<td>6960</td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>2720</td>
<td>3360</td>
<td>5024</td>
<td>5413</td>
</tr>
<tr>
<td></td>
<td>t</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>119</td>
</tr>
<tr>
<td>BIKE</td>
<td>n</td>
<td>24646</td>
<td>49318</td>
<td>81946</td>
<td></td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>12323</td>
<td>24659</td>
<td>40973</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t</td>
<td>134</td>
<td>199</td>
<td>264</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t</td>
<td>142</td>
<td>206</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>HQC</td>
<td>n</td>
<td>35338</td>
<td>71702</td>
<td>115274</td>
<td></td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>17669</td>
<td>35851</td>
<td>57637</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t</td>
<td>132</td>
<td>200</td>
<td>262</td>
<td></td>
</tr>
</tbody>
</table>

Remark 1. Table 1 compares the bit-security levels between our algorithms and [20] for Classic McEliece in $2^{60}$-bit limitation for memory and we provide the following observations.

a) In constant memory and cube-root memory penalty, for all parameters, our algorithms always have lower bit-security levels than all results in [20, 22, 23], and in particular, the best result is 11.09 bits lower than that of [20] for $n = 8192$ in cube-root memory penalty model.

b) For the scenario of logarithmic memory penalty, our algorithms only have the improvements of the bit-security levels for the parameter sets of the categories 1 and 3 in [20].

c) In constant memory, for parameters $n = 4608$, $n = 6688$ and $n = 6960$, the bit complexity of [20] are 187, 262, and 263, respectively, but our results are 179.44, 253.71, and 254.21, respectively. Obviously, our bit-complexity for these three parameters do not achieve the claimed security levels, while [20] show that only $n = 4608$ does not achieve the claimed security levels.

Remark 2. Table 2 compares the bit-security levels between our algorithms and [20] for HQC and BIKE in $2^{60}$-bit limitation for memory and we give some considerations as follows.
a) For the parameters of HQC and BIKE, V1 of the four algorithm instantiations in this paper always performs the best, and so we only list the bit-security levels for the instantiation V1.

b) In three memory models, our bit-security levels are always lower than those of [20] for all parameters of both HQC and BIKE.

5. Conclusion

In this paper, we apply four types of 4-list sum algorithms to the ISD framework to derive lower time complexity in low memory. Furthermore, the new bit-security levels of the Classic McEliece, HQC and BIKE schemes are determined by using our improved ISD algorithms under three memory models, i.e., constant memory, logarithmic memory penalty and cube-root memory penalty, respectively, and we compare these bit-security levels with the latest results in [20]. The numerical results show that for the parameter sets of the Classic McEliece, HQC and BIKE schemes, our algorithms always have lower bit-security levels than [20] in $2^{60}$-bit limitation for memory. Therefore, our results provide more possibilities for security estimation of the code-based cryptographic schemes and are indicative in the design and implementation of the schemes in the future.

References


