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

Yu Li[a,b], Li-Ping Wang[a,b,*]

[a]*State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China*
[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 technique 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 theoretically reestimate the security levels of the parameter sets suggested by these three schemes in low memory by applying K-list sum algorithms to ISD framework. Compared with Esser-Bellini's algorithms, our results achieve the best gains for Classic McEliece, HQC, and BIKE, with reductions in 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

*Corresponding author. E-mail address: wangliping@iie.ac.cn

to the 4th round of the process, three are code-based schemes, namely Classic McEliece, HQC, and BIKE [1]. Currently, a crucial task is to further analyze the 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 [2, 3]. 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 [4]. 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 [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], in which the major ISD algorithms are Stern [18], MMT [15], BJMM [6], May-Ozerov [16] and Both-May [8, 20]. 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, advanced 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 [19]. 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 [21]. Additionally, Esser, May, and Zweydinger 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

[22]. However, advanced ISD 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 [23]. Guo et al. also obtained the bit-security levels for these schemes by proposing a new ISD algorithm [24]. 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 [25] to ISD algorithms.

To be precise, "low memory" here means that the memory consumption of the ISD algorithm is limited to a low upper bound, such as no more than 60 bits, when estimating the actual security levels of the corresponding schemes. This is because advanced ISD algorithms tend to consume high memory when obtaining the lowest security level for a given scheme, and such memory is not implemented in practice. However, if the memory is limited to a certain low upper bound, then the security levels of the corresponding schemes increases dramatically. Table 1 presents the estimation of security levels for Classic McEliece by using different ISD algorithms from [21]. For Category 5c, May-Ozerov's algorithm consumes up to 194 bits of memory in order to obtain a security level of 276 bits. If memory consumption is limited to no more than 60 bits, then the security level increases to 298 bits from 276 bits. Therefore, this motivates us to obtain lower security levels than [21] under the condition that the memory does not exceed 60 bits by applying K-list sum algorithms to ISD framework.

Table 1: Bit-security levels of Classic McEliece by [21].

	Category 1 ($n = 3488$)		Category 3 ($n = 4608$)		Category 5a ($n = 6688$)		Category 5b ($n = 6960$)		Category 5c ($n = 8192$)	
	T	M	T	M	T	M	T	M	T	M
Prange	173	22	217	23	296	24	297	24	334	24
Stern	151	50	193	60	268	80	268	90	303	109
Both-May	143	88	182	101	250	136	249	137	281	141
BJMM	142	97	183	121	248	160	248	163	278	189
May-Ozerov	141	89	180	113	246	165	246	160	276	194
$M \leq 60$	145	60	187	60	262	58	263	60	298	59

* $M \leq 60$ denotes the lowest security level obtained by the above five algorithms when the memory does not exceed 60 bits.

The solving algorithm for a K-list sum problem is essentially a general birthday algorithm and first proposed by Wagner in 2002 [26], 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 [27], while Becker et al. focused on the scenario of $K = 16$ [28]. 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 [29]. In 2019, Dinur unified all previous improvements and analyses of the general birthday algorithms into an algebraic framework [25]. 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 [30] and the learning parity with noise (LPN) problem [31]. By using K-list sum algorithms in the ISD framework, Bricout et al. [32] reselected the parameter sets of the Wave scheme, a code-based digital signature scheme on a ternary field [33], to meet given security levels. Karpman-Lefevre [34] 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 [35]. Additionally, Wang and Liu improved the ISD algorithm under restricted memory by utilizing the K-dissection algorithms proposed in [29] [36]. 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 [25] 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 security levels for the Classic McEliece, HQC, and BIKE schemes in theory. Then, we compare our results with the latest results in [21]. Our best results for the security levels of Classic McEliece, HQC, and BIKE schemes, under a 60-bit limitation for memory, are 11.09, 12.64, and 12.19 bits lower than those in [21], 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 [21] show that only the parameter set for $n = 4608$ fails to achieve the claimed security level.

Organization. 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 [21]. 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 $\{1, 2, \cdots, n\}$ 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 $wt(\mathbf{v})$. Given a length-n vector $\mathbf{v}$ and a matrix $\mathbf{M}$ with n columns, $\mathbf{v}_S$ denotes the vector composed of the terms of $\mathbf{v}$ indexed by S, and $\mathbf{M}_S$ 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 $\delta_{S,[n]}$: $\mathbb{F}_2^{|S|} \to \mathbb{F}_2^n$ by $\delta_{S,[n]}(\mathbf{u}) = \mathbf{v}$ such that $\mathbf{v}_S = \mathbf{u}$ and $\mathbf{v}_{[n]\backslash S} = \mathbf{0}$, 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 $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ form a basis of an $[n, k]$ code $\mathcal{C}$, then the matrix $\mathbf{G}$ is called as a generator matrix of code $\mathcal{C}$. Meanwhile, a matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ is a parity-check matrix of $\mathcal{C}$ if its kernel corresponds to an $[n, k]$ code $\mathcal{C}$.*

Definition 3. *A syndrome $\mathbf{s}$ of a random vector $\mathbf{v} \in \mathbb{F}_2^n$ is obtained as $\mathbf{s} = \mathbf{v}\mathbf{H}^\top$, where $\mathbf{H}$ is a parity-check matrix of an $[n, k]$ code, and one has $\mathbf{v} \in \mathcal{C}$ if and only if $\mathbf{v}\mathbf{H}^\top = 0$.*

Problem 1 ((n,k,t)-SD problem)**.** *Input a parity-check matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *of an* $[n,k]$ *linear code* $\mathcal{C}$*, a syndrome* $\mathbf{s} \in \mathbb{F}_2^{n-k}$ *and a given positive integer* t*, output an error vector* $\mathbf{e} \in \mathbb{F}_2^n$ *such that* $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$ *and* $wt_H(\mathbf{e}) = t$*.*

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 [32] and the checkable multiple syndrome decoding problem in [35].

Problem 2 ($(n,n-k,t,m)$-MSSD problem)**.** *Input a parity-check matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *of an* $[n,k]$ *code* $\mathcal{C}$*, a syndrome* $\mathbf{s} \in \mathbb{F}_2^{n-k}$*, and a given positive integer* t*, output a set* $\{\mathbf{e} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top, wt(\mathbf{e}) = t\}$ *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* $\mathcal{C}$ *is a size-*k *subset* $I \subseteq [n]$ *is called if* $|\mathcal{C}_I| = |\mathcal{C}|$*.*

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 $\mathbf{P} \in \mathbb{F}_2^{n\times n}$ such that $(\mathbf{HP})_{[n]\setminus[k+\ell]}$ is partially reversible.

2) **Partial Gaussian elimination step**. Choose an invertible transformation matrix $\mathbf{U} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ satisfying

$$\mathbf{UHP} = \begin{pmatrix} \widehat{\mathbf{H}} & \mathbf{0} \\ \widetilde{\mathbf{H}} & \mathbf{I}_{n-k-\ell} \end{pmatrix}, \quad \mathbf{U}\mathbf{s}^\top = \begin{pmatrix} \hat{\mathbf{s}} \\ \tilde{\mathbf{s}} \end{pmatrix},$$

where $\widehat{\mathbf{H}} \in \mathbb{F}_2^{\ell\times(k+\ell)}$, $\widetilde{\mathbf{H}} \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}$, $\hat{\mathbf{s}} \in \mathbb{F}_2^{\ell}$ and $\tilde{\mathbf{s}} \in \mathbb{F}_2^{n-k-\ell}$. Meanwhile, the error vector $\mathbf{e}$ is split into two vectors $\hat{\mathbf{e}} \in \mathbb{F}_2^{k+\ell}$ and $\tilde{\mathbf{e}} \in \mathbb{F}_2^{n-k-\ell}$, i.e., $\mathbf{eP} = (\hat{\mathbf{e}}^\top|\tilde{\mathbf{e}}^\top)$. Thus, we have

$$\widehat{\mathbf{H}}\hat{\mathbf{e}} = \hat{\mathbf{s}}, \tag{1}$$

$$\widetilde{\mathbf{H}}\hat{\mathbf{e}} + \tilde{\mathbf{e}} = \tilde{\mathbf{s}}. \tag{2}$$

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

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

The above framework is an iterative structure where each iteration is to find 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 [35], the probability of success in step 4) is

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

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

Lemma 1 ([35]). *For positive integer parameters n, k, t, ℓ, 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*

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

where P_{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 $\big\{(\mathbf{y}_1, \ldots, \mathbf{y}_K) \in L_1 \times \cdots \times L_K : \mathbf{y}_1 + \cdots + \mathbf{y}_K = 0\big\}$.*

In this paper, we consider a solving algorithm of a 4-list sum problem over $\mathbb{F}_2$. In [25], author proposed a 4-list sum algorithm, where the underlying lists have the equal sizes, i.e., $|L_1| = |L_2| = |L_3| = |L_4|$. Here, we general the algorithm by setting

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

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 $L_{1234} = \{(\mathbf{x}_1, \ldots, \mathbf{x}_4) \in L_1 \times \cdots \times L_4 : \mathbf{x}_1 + \cdots + \mathbf{x}_4 = 0\}$.

Initialize $L_{12}, L_{34}, L_{1234} \leftarrow \emptyset$.
for all $(\mathbf{0}|\mathbf{x}) \in 0^{m_2-v} \times \mathbb{F}_2^v$ **do**
  for all $\mathbf{x}_1 \in L_1$ **do**
    $L_{12} \leftarrow L_{12} \cup \{\mathbf{x}_1 + \mathbf{x}_2 : (\mathbf{x}_1 + \mathbf{x}_2)_{[m_2]} = (\mathbf{0}|\mathbf{x}), \mathbf{x}_2 \in L_2\}$
  end for
  for all $\mathbf{x}_3 \in L_3$ **do**
    $L_{34} \leftarrow L_{34} \cup \{\mathbf{x}_3 + \mathbf{x}_4 : (\mathbf{x}_3 + \mathbf{x}_4)_{[m_2]} = (\mathbf{0}|\mathbf{x}), \mathbf{x}_4 \in L_4\}$
  end for
  for all $\mathbf{x}_{12} \in L_{12}$ **do**
    $L_{1234} \leftarrow L_{1234} \cup \{\mathbf{x}_{12} + \mathbf{x}_{34} : (\mathbf{x}_{12} + \mathbf{x}_{34})_{[n]\setminus[m_2]} = \mathbf{0}, \mathbf{x}_{34} \in L_{34}\}$
  end for
end for

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

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 technique 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.,*

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

Proof. Since the elements in the sets L_1 and L_2 are randomly uniform and

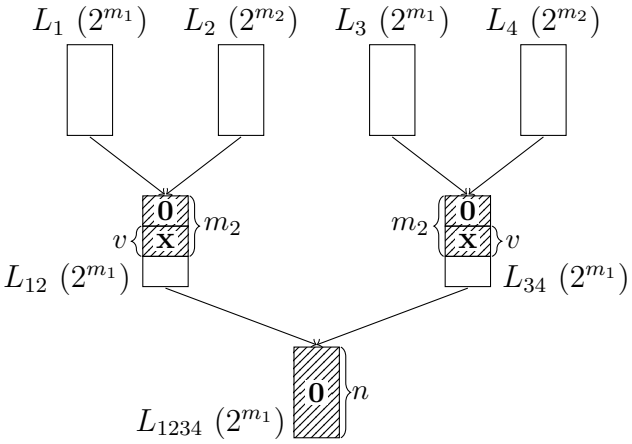

Figure 1: 4-list sum algorithm with memory 2^{m_1} and length n with $m_1 \geq m_2$.

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

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:

$$\widehat{\mathbf{H}}\hat{\mathbf{e}} = \hat{\mathbf{s}}, \tag{3}$$

$$\widetilde{\mathbf{H}}\hat{\mathbf{e}} + \tilde{\mathbf{e}} = \tilde{\mathbf{s}}. \tag{4}$$

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

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

$$\begin{aligned}\widehat{\mathbf{H}}\hat{\mathbf{e}} &= \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_4 \end{pmatrix} \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \mathbf{e}_4 \end{pmatrix} \\ &= \mathbf{H}_1\mathbf{e}_1 + \mathbf{H}_2\mathbf{e}_2 + \mathbf{H}_3\mathbf{e}_3 + \mathbf{H}_4\mathbf{e}_4 = \hat{\mathbf{s}},\end{aligned}$$

and then construct the four sets as follows:

$$L_i = \Big\{(\mathbf{e}_i, \mathbf{H}_i\mathbf{e}_i) : \mathbf{H}_i \in \mathbb{F}_2^{\ell\times n_i}, \mathbf{e}_i \in \mathbb{F}_2^{n_i}, wt(\mathbf{e}_i) = p_i\Big\}$$

for $i = 1, 2, 3$ and

$$L_4 = \Big\{(\mathbf{e}_4, \mathbf{H}_4\mathbf{e}_4 + \hat{\mathbf{s}}) : \mathbf{H}_i \in \mathbb{F}_2^{\ell\times n_4}, \mathbf{e}_4 \in \mathbb{F}_2^{n_4}, wt(\mathbf{e}_4) = p_4\Big\},$$

where $n_1 + n_2 + n_3 + n_4 = k + \ell$, $p_1 + p_2 + p_3 + p_3 + p_4 = p$ 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 $(\mathbf{H}_1\mathbf{e}_1, \mathbf{H}_2\mathbf{e}_2, \mathbf{H}_3\mathbf{e}_3, \mathbf{H}_4\mathbf{e}_4)$ satisfying the condition $\mathbf{H}_1\mathbf{e}_1 + \mathbf{H}_2\mathbf{e}_2 + \mathbf{H}_3\mathbf{e}_3 + \mathbf{H}_4\mathbf{e}_4 + \hat{\mathbf{s}} = \mathbf{0}$ is found, record and store the corresponding 4-tuple $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)$. 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 $\hat{\mathbf{e}} = (\mathbf{e}_1^\top | \mathbf{e}_2^\top | \mathbf{e}_3^\top | \mathbf{e}_4^\top)$ satisfying Equation (3).

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.

3.2. Analysis of complexity

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} and memory consumption M_{ISD}, where*

$$\begin{aligned} T_{ISD} &= O\Bigg(\max\{(n+1)(n-k)^2, 2^{m_1+v}\} \\ &\qquad \cdot \max\left\{1, \frac{\max\left\{1, \min\left\{\binom{n}{t} \cdot 2^{-\ell}, 2^{n-k-\ell}\right\}\right\}}{\binom{n-k-\ell}{t-p} \cdot 2^{m_1}}\right\}\Bigg), \\ M_{ISD} &= O\big(\max\{n(n-k), 2^{m_1}\}\big), \end{aligned}$$

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

Algorithm 2 (ISD by 4-list sum algorithm)

Input: a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_2^{n-k}$ and the parameters $t, m_1, \ell, v, p_i, n_i$ for $i = 1, 2, 3, 4$, which satisfy $m_1 := \binom{n_1}{p_1} = \binom{n_3}{p_3} \geq \binom{n_2}{p_2} = \binom{n_4}{p_4} := m_2$ and $\ell = m_1 + m_2 + v$.

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

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

$$\mathbf{U}\mathbf{H}_I = \begin{pmatrix} \widehat{\mathbf{H}} \\ \widetilde{\mathbf{H}} \end{pmatrix}, \ \mathbf{U}\mathbf{H}_{[n]\setminus I} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_{n-K-\ell} \end{pmatrix}, \mathbf{U}\mathbf{s}^\top = \begin{pmatrix} \hat{\mathbf{s}} \\ \tilde{\mathbf{s}} \end{pmatrix},$$

where $\widehat{\mathbf{H}} \in \mathbb{F}_2^{\ell\times(k+\ell)}$, $\widetilde{\mathbf{H}} \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}$, $\hat{\mathbf{s}} \in \mathbb{F}_2^{\ell}$, $\tilde{\mathbf{s}} \in \mathbb{F}_2^{n-k-\ell}$.
Let $\mathbf{H}_i = (\widehat{\mathbf{H}})_{I_i}$ for $i = 1, 2, 3, 4$. Build four sets

$$L_i = \Big\{(\mathbf{e}_i, \mathbf{H}_i\mathbf{e}_i) : \mathbf{H}_i \in \mathbb{F}_2^{\ell\times n_i}, \mathbf{e}_i \in \mathbb{F}_2^{n_i}, wt(\mathbf{e}_1) = p_i\Big\}, \quad i = 1, 2, 3,$$
$$L_4 = \Big\{(\mathbf{e}_4, \mathbf{H}_4\mathbf{e}_4 + \hat{\mathbf{s}}) : \mathbf{H}_4 \in \mathbb{F}_2^{\ell\times n_4}, \mathbf{e}_4 \in \mathbb{F}_2^{n_4}, wt(\mathbf{e}_1) = p_4\Big\}.$$

Initialize $L_{12}, L_{34}, L_{1234} \leftarrow \emptyset$.
for all $(\mathbf{0}|\mathbf{y}) \in 0^{m_2-v} \times \mathbb{F}_2^v$ **do**
  for all $(\mathbf{e}_1, \mathbf{y}_1) \in L_1$ **do**
    $L_{12} \leftarrow L_{12} \cup \{(\mathbf{e}_{12}, \mathbf{H}_{12}\mathbf{e}_{12}) : I_{12} = I_1 \cup I_2, \ \mathbf{e}_{12} = \delta_{I_1,I_{12}}(\mathbf{e}_1) + \delta_{I_2,I_{12}}(\mathbf{e}_2), \ \mathbf{H}_{12} = \widehat{\mathbf{H}}_{I_{12}}, \ (\mathbf{y}_1 + \mathbf{y}_2)_{[m_2]} = (\mathbf{0}|\mathbf{y}), \ (\mathbf{e}_2, \mathbf{y}_2) \in L_2\}$.
  end for
  for all $(\mathbf{e}_3, \mathbf{y}_3) \in L_3$ **do**
    $L_{34} \leftarrow L_{34} \cup \{(\mathbf{e}_{34}, \mathbf{H}_{34}\mathbf{e}_{34} + \hat{\mathbf{s}}) : I_{34} = I_3 \cup I_4, \ \mathbf{H}_{34} = \widehat{\mathbf{H}}_{I_{34}}, \ \mathbf{e}_{34} = \delta_{I_3,I_{34}}(\mathbf{e}_1) + \delta_{I_4,I_{34}}(\mathbf{e}_4), \ (\mathbf{y}_3 + \mathbf{y}_4)_{[m_2]} = (\mathbf{0}|\mathbf{y}), \ (\mathbf{e}_4, \mathbf{y}_4) \in L_4\}$.
  end for
  for all $(\mathbf{e}_{12}, \mathbf{y}_{12}) \in L_{12}$ **do**
    $L_{1234} \leftarrow L_{1234} \cup \{\hat{\mathbf{e}} = \delta_{I_{12},I}(\mathbf{e}_{12}) + \delta_{I_{34},I}(\mathbf{e}_{34}) : (\mathbf{y}_{12} + \mathbf{y}_{34})_{[\ell]\setminus[m_2]} = \mathbf{0}, (\mathbf{e}_{34}, \mathbf{y}_{34}) \in L_{34}\}$.
  end for
end for
for all $\hat{\mathbf{e}} \in L_{1234}$ **do**
  if $wt(\tilde{\mathbf{s}} - \widetilde{\mathbf{H}}\hat{\mathbf{e}})) = t - (p_1 + p_2 + p_3 + p_4)$ **then**
    Output: $\mathbf{e} = \delta_{I,[n]}(\hat{\mathbf{e}}) + \delta_{[n]\setminus I,[n]}\big(\tilde{\mathbf{s}} - \widetilde{\mathbf{H}}\hat{\mathbf{e}}\big)$.
  else
    Return to Step 1.
  end if
end for

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

$$T_{MSSD} = 2^{m_1+v}, M_{MSSD} = 2^{m_1}.$$

In addition, the memory of the parity-check matrix is $n(n-k)$ and hence the result is obtained according to Lemma 1. □

Next, we give four instantiations of Theorem 1 by choosing different parameters m_1 and m_2 as follows:

- Version 1 (V1): $\binom{\lfloor (k+\ell)/4 \rceil}{p/4} = 2^{m_1} = 2^{m_2}$,
- Version 2 (V2): $\binom{\lfloor (k+\ell)/3 \rceil}{p/3} = 2^{m_1}$, $\binom{\lfloor (k+\ell)/6 \rceil}{p/6} = 2^{m_2}$,
- Version 3 (V3): $\binom{\lfloor 2(k+\ell)/5 \rceil}{2p/5} = 2^{m_1}$, $\binom{\lfloor (k+\ell)/10 \rceil}{p/10} = 2^{m_2}$,
- Version 4 (V4): $\binom{\lfloor 3(k+\ell)/7 \rceil}{3p/7} = 2^{m_1}$, $\binom{\lfloor (k+\ell)/14 \rceil}{p/14} = 2^{m_2}$,

where ℓ needs to satisfy $\ell = m_1 + m_2 + v$ according to Theorem 1.

4. Numerical Results

In this section, we use 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 [37] and [22] 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 [21], 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, [21] and [23] 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, "V_1", "V_2", "V_3" and "V_4" denote four instantiations of 4-list sum algorithms in Section 3.2, respectively, and "$M \leq 60$" indicates that the finial 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 [21], the values in parentheses refer to the gain of the bit-security levels obtained by our algorithms relative to that in [21], i.e., $T_{[21]} - T_{\text{ours}}$.

Remark 1. *Table 2 compares the security levels between our algorithms and [21] for Classic McEliece in 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 [21, 23, 24], and in particular, the best result is 11.09 bits lower than that of [21] for $n = 8192$ *in cube-root memory penalty model.*

b) In constant memory, for parameters $n = 4608$, $n = 6688$ *and* $n = 6960$, *the bit complexity of [21] 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 [21] show that only* $n = 4608$ *does not achieve the claimed security levels.*

c) 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 [21].

Table 2: Bit-security levels of Classic McEliece.

		Category 1		Category 3		Category 5a		Category 5b		Category 5c	
		M	T	M	T	M	T	M	T	M	T
constant:											
	[21]	60	145	60	187	58	262	60	263	59	298
$M \leq 60$	[23]	–	145.47	–	188.16	–	263.16	–	263.64	–	298.65
	[24]	58	143.4	55	184.4	59	259.2	60	259.8	55	296.6
	V1	26	140.04 (4.96)	27	181.22 (5.78)	29	255.02 (6.98)	38	254.21 (8.79)	39	289.07 (8.93)
	V2	35	139.49 (5.51)	37	179.70 (7.30)	39	254.37 (7.63)	39	255.48 (7.52)	58	289.36 (8.64)
	V3	36	138.44 (6.56)	38	179.44 (7.56)	40	255.06 (6.94)	40	255.27 (7.73)	41	291.06 (6.94)
	V4	52	139.71 (5.29)	54	180.88 (6.12)	58	253.71 (8.29)	58	254.83 (8.17)	60	289.71 (8.29)
logarithmic:											
	[21]	89	147	113	187	165	253	160	253	194	283
$M \leq 60$	[23]	–	151.06	–	193.59	–	268.66	–	269.17	–	304.19
	V1	26	144.74 (2.26)	27	185.97 (1.03)	29	259.88 (-6.88)	38	259.46 (-6.46)	39	294.36 (-11.36)
	V2	35	144.62 (2.38)	37	184.90 (2.10)	39	259.66 (-6.66)	39	260.77 (-7.77)	58	295.21 (-12.21)
	V3	36	143.61 (3.39)	38	184.69 (2.31)	40	260.38 (-7.38)	40	260.59 (-7.59)	41	296.42 (-13.43)
	V4	52	145.41 (1.59)	54	186.64 (0.36)	58	259.57 (-6.57)	58	160.69 (-7.69)	60	295.62 (-12.62)
cube-root:											
	[21]	25	156	26	199	36	275	36	276	47	312
	[23]	–	157.25	–	199.69	–	276.13	–	276.97	–	313.12
	V1	18	148.05 (7.95)	23	189.32 (9.68)	29	264.69 (10.31)	29	265.69 (10.31)	30	300.91 (11.09)
	V2	35	151.16 (4.84)	37	192.03 (6.97)	39	267.37 (7.63)	39	268.48 (7.52)	40	303.71 (8.29)
	V3	36	150.44 (5.56)	38	192.11 (6.89)	40	268.39 (6.61)	40	268.60 (7.50)	41	304.73 (7.27)
	V4	52	157.05 (-1.05)	54	198.88 (0.12)	58	273.05 (1.95)	58	274.17 (1.83)	60	309.71 (2.29)

Table 3: Bit-security levels of HQC and BIKE.

		Category 1		Category 3		Category 5	
		M	T	M	T	M	T
constant: $M \leq 61$							
BIKE-message	[21]	40	146	43	211	61	276
BIKE-message	[23]	–	142.35	–	206.41	–	272.26
BIKE-message	[24]	46	134.7	50	198.3	53	262.4
BIKE-message	V1	29	135.07 (10.93)	31	198.81 (12.19)	32	263.92 (12.08)
BIKE-key	[21]	40	147	57	210	61	278
BIKE-key	[23]	–	143.73	–	205.93	–	274.13
BIKE-key	[24]	46	140.7	50	203.6	53	270.6
BIKE-key	V1	29	142.50 (4.50)	31	205.47 (4.53)	32	273.55 (4.45)
HQC	[21]	39	145	44	213	39	276
HQC	[23]	–	141.16	–	208.19	–	271.14
HQC	[24]	48	133.6	52	200.1	55	261.2
HQC	V1	30	134.74 (10.26)	32	201.43 (11.57)	27	263.36 (12.64)
logarithmic:							
BIKE-message	[21]	31	150	33	215	34	281
BIKE-message	[23]	–	142.64	–	206.79	–	272.84
BIKE-message	V1	29	139.93 (10.07)	31	203.76 (11.24)	32	268.92 (12.08)
BIKE-key	[21]	31	151	33	215	34	283
BIKE-key	[23]	–	144.24	–	206.90	–	275.50
BIKE-key	V1	29	147.36 (3.64)	31	210.42 (4.58)	32	278.55 (4.45)
HQC	[21]	32	150	34	218	36	280
HQC	[23]	–	141.49	–	208.61	–	271.62
HQC	V1	30	139.65 (10.35)	32	206.43 (11.57)	27	268.40 (11.60)
cube-root:							
BIKE-message	[21]	30	152	32	217	33	283
BIKE-message	[23]	–	143.37	–	207.77	–	273.99
BIKE-message	V1	29	144.74 (7.26)	31	209.14 (7.86)	32	274.59 (8.41)
BIKE-key	[21]	30	153	32	217	33	285
BIKE-key	[23]	–	144.89	–	207.79	–	276.60
BIKE-key	V1	29	152.17 (0.83)	31	215.80 (1.2)	32	284.22 (0.78)
HQC	[21]	31	151	33	220	35	282
HQC	[23]	–	142.33	–	209.71	–	272.90
HQC	V1	30	144.74 (6.26)	32	212.10 (7.90)	27	274.36 (7.64)

Table 4: The parameters in Table 2.

	Category 1 (ℓ, p, v)	Category 3 (ℓ, p, v)	Category 5a (ℓ, p, v)	Category 5b (ℓ, p, v)	Category 5c (ℓ, p, v)
constant:					
V1	$(57, 12, 5)$	$(60, 12, 6)$	$(63, 12, 5)$	$(76, 16, 0)$	$(78, 16, 0)$
V2	$(52, 12, 0)$	$(55, 12, 0)$	$(58, 12, 0)$	$(58, 12, 0)$	$(86, 18, 0)$
V3	$(45, 10, 0)$	$(47, 10, 0)$	$(49, 10, 0)$	$(50, 10, 0)$	$(51, 10, 0)$
V4	$(60, 14, 0)$	$(62, 14, 0)$	$(67, 14, 0)$	$(67, 14, 0)$	$(69, 14, 0)$
logarithmic:					
V1	$(57, 12, 5)$	$(60, 12, 6)$	$(63, 12, 5)$	$(76, 16, 0)$	$(78, 16, 0)$
V2	$(52, 12, 0)$	$(55, 12, 0)$	$(58, 12, 0)$	$(58, 12, 0)$	$(86, 18, 0)$
V3	$(45, 10, 0)$	$(47, 10, 0)$	$(49, 10, 0)$	$(50, 10, 0)$	$(51, 10, 0)$
V4	$(60, 14, 0)$	$(62, 14, 0)$	$(67, 14, 0)$	$(67, 14, 0)$	$(69, 14, 0)$
cube-root:					
V1	$(49, 8, 13)$	$(52, 8, 14)$	$(63, 12, 5)$	$(63, 12, 5)$	$(65, 12, 5)$
V2	$(52, 12, 0)$	$(55, 12, 0)$	$(58, 12, 0)$	$(58, 12, 0)$	$(60, 12, 0)$
V3	$(45, 10, 0)$	$(47, 10, 0)$	$(49, 10, 0)$	$(50, 10, 0)$	$(51, 10, 0)$
V4	$(60, 14, 0)$	$(62, 14, 0)$	$(67, 14, 0)$	$(67, 14, 0)$	$(69, 14, 0)$

Table 5: The parameters in Table 3.

		Category 1 (ℓ, p, v)	Category 3 (ℓ, p, v)	Category 5 (ℓ, p, v)
constant:				
BIKE-message	V1	$(68, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
BIKE-key	V1	$(65, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
HQC	V1	$(68, 8, 20)$	$(73, 8, 21)$	$(76, 8, 22)$
logarithmic:				
BIKE-message	V1	$(68, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
BIKE-key	V1	$(65, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
HQC	V1	$(68, 8, 20)$	$(73, 8, 21)$	$(76, 8, 22)$
cube-root:				
BIKE-message	V1	$(68, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
BIKE-key	V1	$(65, 8, 19)$	$(70, 8, 20)$	$(73, 8, 21)$
HQC	V1	$(68, 8, 20)$	$(73, 8, 21)$	$(76, 8, 22)$

Table 6: Security parameters of Classic McEliece, HQC and BIKE.

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

Remark 2. *Table 3 compares the security levels between our algorithms and [21] for HQC and BIKE in* 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 [21] 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 security levels of the Classic McEliece, HQC and BIKE schemes are determined by using our ISD algorithms under three memory models, i.e., constant memory, logarithmic memory penalty and cube-root memory penalty, respectively, and we compare these security levels with the latest results in [21]. The numerical results show that for the parameter sets of the Classic McEliece, HQC and BIKE schemes, our algorithms always have lower security levels than [21] in 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

[1] L. Chen, D. Moody, Y.-K. Liu, Post-quantum cryptography, NIST (2022). URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[2] S. Barg, Some new np-complete coding problems, Problemy Peredachi Informatsii 30 (1994) 23–28. URL: http://www.mathnet.ru/links/0d9ea50bc1746a48c28b580b628bbaf1/ppi241.pdf.

[3] E. R. Berlekamp, R. J. McEliece, H. C. A. van Tilborg, On the inherent intractability of certain coding problems (corresp.), IEEE Trans. Inf. Theory 24 (1978) 384–386. URL: https://doi.org/10.1109/TIT.1978.1055873.

[4] E. Prange, The use of information sets in decoding cyclic codes, IRE Trans. Inf. Theory 8 (1962) 5–9. URL: https://doi.org/10.1109/TIT.1962.1057777.

[5] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, P. Santini, A finite regime analysis of information set decoding algorithms, Algorithms 12 (2019) 209. URL: https://doi.org/10.3390/a12100209.

[6] A. Becker, A. Joux, A. May, A. Meurer, Decoding random binary linear codes in $2^{n/20}$: How 1+1=0 improves information set decoding, in: D. Pointcheval, T. Johansson (Eds.), EUROCRYPT 2012, volume 7237 of *LNCS*, Springer, 2012, pp. 520–536. URL: https://doi.org/10.1007/978-3-642-29011-4_31.

[7] D. J. Bernstein, T. Lange, C. Peters, Smaller decoding exponents: Ball-collision decoding, in: P. Rogaway (Ed.), CRYPTO 2011, volume 6841 of *LNCS*, Springer, 2011, pp. 743–760. URL: https://doi.org/10.1007/978-3-642-22792-9_42.

[8] L. Both, A. May, Decoding linear codes with high error rate and its impact for LPN security, in: T. Lange, R. Steinwandt (Eds.), PQCrypto 2018, volume 10786 of *LNCS*, Springer, 2018, pp. 25–46. URL: https://doi.org/10.1007/978-3-319-79063-3_2.

[9] A. Canteaut, F. Chabaud, A new algorithm for finding minimum-weight words in a linear code: Application to mceliece's cryptosystem and to narrow-sense BCH codes of length 511, IEEE Trans. Inf. Theory 44 (1998) 367–378. URL: https://doi.org/10.1109/18.651067.

[10] A. Canteaut, N. Sendrier, Cryptanalysis of the original mceliece cryptosystem, in: K. Ohta, D. Pei (Eds.), ASIACRYPT 1998, volume 1514 of *LNCS*, Springer, 1998, pp. 187–199. URL: https://doi.org/10.1007/3-540-49649-1_16.

[11] I. I. Dumer, Two decoding algorithms for linear codes, Probl. Peredachi Inf. 25 (1989) 24–32. URL: http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=635&option_lang=eng.

[12] M. Finiasz, N. Sendrier, Security bounds for the design of code-based cryptosystems, in: M. Matsui (Ed.), ASIACRYPT 2009, volume 5912 of *LNCS*, Springer, 2009, pp. 88–105. URL: https://doi.org/10.1007/978-3-642-10366-7_6.

[13] P. J. Lee, E. F. Brickell, An observation on the security of mceliece's public-key cryptosystem, in: C. G. Günther (Ed.), EUROCRYPT 1988, volume 330 of *LNCS*, Springer, 1988, pp. 275–280. URL: https://doi.org/10.1007/3-540-45961-8_25.

[14] J. S. Leon, A probabilistic algorithm for computing minimum weights of large error-correcting codes, IEEE Trans. Inf. Theory 34 (1988) 1354–1359. URL: https://doi.org/10.1109/18.21270.

[15] A. May, A. Meurer, E. Thomae, Decoding random linear codes in $o(2^{0.054n})$, in: D. H. Lee, X. Wang (Eds.), ASIACRYPT 2011, volume 7073 of *LNCS*, Springer, 2011, pp. 107–124. URL: https://doi.org/10.1007/978-3-642-25385-0_6.

[16] A. May, I. Ozerov, On computing nearest neighbors with applications to decoding of binary linear codes, in: E. Oswald, M. Fischlin (Eds.), EUROCRYPT 2015, Part I, volume 9056 of *LNCS*, Springer, 2015, pp. 203–228. URL: https://doi.org/10.1007/978-3-662-46800-5_9.

[17] N. Sendrier, Decoding one out of many, in: B. Yang (Ed.), PQCrypto 2011, volume 7071 of *LNCS*, Springer, 2011, pp. 51–

67. URL: https://doi.org/10.1007/978-3-642-25405-5_4. doi:10.1007/978-3-642-25405-5_4.

[18] J. Stern, A method for finding codewords of small weight, in: G. D. Cohen, J. Wolfmann (Eds.), Coding Theory and Applications, 3rd International Colloquium, 1988, volume 388 of *Lecture Notes in Computer Science*, Springer, 1988, pp. 106–113. URL: https://doi.org/10.1007/BFb0019850.

[19] R. C. Torres, N. Sendrier, Analysis of information set decoding for a sub-linear error weight, in: T. Takagi (Ed.), PQCrypto 2016, volume 9606 of *LNCS*, Springer, 2016, pp. 144–161. URL: https://doi.org/10.1007/978-3-319-29360-8_10.

[20] A. Esser, Revisiting nearest-neighbor-based information set decoding, IACR Cryptol. ePrint Arch. (2022) 1328. URL: https://eprint.iacr.org/2022/1328.

[21] A. Esser, E. Bellini, Syndrome decoding estimator, in: G. Hanaoka, J. Shikata, Y. Watanabe (Eds.), PKC 2022, Part I, volume 13177 of *LNCS*, Springer, 2022, pp. 112–141. URL: https://doi.org/10.1007/978-3-030-97121-2_5.

[22] A. Esser, A. May, F. Zweydinger, Mceliece needs a break - solving mceliece-1284 and quasi-cyclic-2918 with modern ISD, in: O. Dunkelman, S. Dziembowski (Eds.), EUROCRYPT 2022, Part III, volume 13277 of *LNCS*, Springer, 2022, pp. 433–457. URL: https://doi.org/10.1007/978-3-031-07082-2_16.

[23] A. Esser, F. Zweydinger, New time-memory trade-offs for subset sum - improving ISD in theory and practice, IACR Cryptol. ePrint Arch. (2022) 1329. URL: https://eprint.iacr.org/2022/1329.

[24] Q. Guo, T. Johansson, V. Nguyen, A new sieving-style information-set decoding algorithm, IACR Cryptol. ePrint Arch. (2023) 247. URL: https://eprint.iacr.org/2023/247.

[25] I. Dinur, An algorithmic framework for the generalized birthday problem, Des. Codes Cryptogr. 87 (2019) 1897–1926. URL: https://doi.org/10.1007/s10623-018-00594-6.

[26] D. A. Wagner, A generalized birthday problem, in: M. Yung (Ed.), CRYPTO 2002, volume 2442 of *LNCS*, Springer, 2002, pp. 288–303. URL: https://doi.org/10.1007/3-540-45708-9_19.

[27] A. Horlemann, S. Puchinger, J. Renner, T. Schamberger, A. Wachter-Zeh, Information-set decoding with hints, in: A. Wachter-Zeh, H. Bartz, G. Liva (Eds.), CBCrypto 2021, volume 13150 of *LNCS*, Springer, 2021, pp. 60–83. URL: https://doi.org/10.1007/978-3-030-98365-9_4.

[28] A. Becker, J. Coron, A. Joux, Improved generic algorithms for hard knapsacks, in: K. G. Paterson (Ed.), EUROCRYPT 2011, volume 6632 of *LNCS*, Springer, 2011, pp. 364–385. URL: https://doi.org/10.1007/978-3-642-20465-4_21.

[29] I. Dinur, O. Dunkelman, N. Keller, A. Shamir, Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems, in: R. Safavi-Naini, R. Canetti (Eds.), CRYPTO 2012, volume 7417 of *LNCS*, Springer, 2012, pp. 719–740. URL: https://doi.org/10.1007/978-3-642-32009-5_42.

[30] N. Howgrave-Graham, A. Joux, New generic algorithms for hard knapsacks, in: H. Gilbert (Ed.), EUROCRYPT 2010, volume 6110 of *LNCS*, Springer, 2010, pp. 235–256. URL: https://doi.org/10.1007/978-3-642-13190-5_12.

[31] A. Esser, F. Heuer, R. Kübler, A. May, C. Sohler, Dissection-bkw, in: H. Shacham, A. Boldyreva (Eds.), CRYPTO 2018, Part II, volume 10992 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 638–666. URL: https://doi.org/10.1007/978-3-319-96881-0_22.

[32] R. Bricout, A. Chailloux, T. Debris-Alazard, M. Lequesne, Ternary syndrome decoding with large weight, in: K. G. Paterson, D. Stebila (Eds.), SAC 2019, volume 11959 of *LNCS*, Springer, 2019, pp. 437–466. URL: https://doi.org/10.1007/978-3-030-38471-5_18.

[33] T. Debris-Alazard, N. Sendrier, J. Tillich, Wave: A new family of trapdoor one-way preimage sampleable functions based on codes, in: S. D. Galbraith, S. Moriai (Eds.), ASIACRYPT 2019, Part I, volume 11921 of *LNCS*, Springer, 2019, pp. 21–51. URL: https://doi.org/10.1007/978-3-030-34578-5_2.

[34] P. Karpman, C. Lefevre, Time-memory tradeoffs for large-weight syndrome decoding in ternary codes, in: G. Hanaoka, J. Shikata, Y. Watanabe (Eds.), PKC 2022, Part I, volume 13177 of *LNCS*, Springer, 2022, pp. 82–111. URL: https://doi.org/10.1007/978-3-030-97121-2_4.

[35] A. Chailloux, T. Debris-Alazard, S. Etinski, Classical and quantum algorithms for generic syndrome decoding problems and applications to the lee metric, in: J. H. Cheon, J. Tillich (Eds.), PQCrypto 2021, volume 12841 of *LNCS*, Springer, 2021, pp. 44–62. URL: https://doi.org/10.1007/978-3-030-81293-5_3.

[36] M. Wang, M. Liu, Improved information set decoding for code-based cryptosystems with constrained memory, in: J. Wang, C. Yap (Eds.), Frontiers in Algorithmics - 9th International Workshop, FAW 2015, volume 9130 of *LNCS*, Springer, 2015, pp. 241–258. URL: https://doi.org/10.1007/978-3-319-19647-3_23.

[37] T. Chou, et al., Classic mceliece: conservative code-based cryptography, 10 October 2020 (2020).