jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 1, PP. 79–85
Manuscript received September 9, 2024; revised March, 2025. doi: 10.24425/ijet.2025.153547

Syndrome decoding with MRHS solver

Miloslav Smičík, and Pavol Zajac

***Abstract*—The syndrome decoding problem (SDP) is an NP-complete problem that has important applications in the development of post-quantum cryptography. Currently, the most efficient algorithms for SDP are based on the Information Set Decoding (ISD) approach that leverages efficiently time-memory and probability trade-offs. In our contribution, we explore a different approach based on transforming an instance of the SDP problem into a so-called Multiple Right-Hand-Sides (MRHS) Equation system. The MRHS system is then solved with a specialized MRHS solver. We explore how difficult is to solve (small) instances of SDP in MRHS form, and which trade-offs and parametric selections lead to the best results. Although our practical results are worse than those obtained by ISD, we believe that they show a better understanding of the connection between SDP and its MRHS representation, and can be a basis for future improvements.**

***Keywords*—Cryptography; Decoding problem; MRHS equation systems**

I. Introduction

ONE of the oldest examples of public key cryptography is the McEliece cryptosystem [1]. Unlike RSA and DLP-based systems, the McEliece cryptosystem is believed to be secure even against the quantum adversary [2]. A modernized version, the Classic McEliece [3], is under consideration for NIST standardization as a post-quantum KEM. A primary difference between the new and old versions is the selection of parameters. The parameters originally proposed by McEliece did not stand the test of time and currently do not provide enough security even against classical adversaries. This development is mainly due to the improvement of the general decoding algorithms [4]. These algorithms try to solve instances of the syndrome decoding problem (SDP), which essentially requires finding a low-weight solution of a specific linear equation system (with more variables than equations). In Section II-A we provide more information about SDP and decoding algorithms.

The most successful algorithms for solving the decoding problem are from a family of information set decoding (ISD) methods. They are highly optimized and use various time-memory-probability trade-offs. It is not known whether these types of algorithms can be improved further, but the research area is still very active. Recently, however, a new type of distinguisher for McEliece cryptosystem based on syzygies was published [5]. The distinguisher uses algebraic properties of the underlying Goppa code and provides a substantial improvement compared to generic ISD methods. Naturally, a question arises as to whether some suitable algebraic methods can be used even for solving general decoding problems.

In [6], [7] we have explored a connection between (regular) decoding problems and the so-called MRHS problem. The MRHS problem asks whether a solution of a set of formal inclusions of type $\mathbf{x}\mathbf{M}_i \in S_i$ (so-called Multiple Right-Hand Sides equations, MRHS system) exists. MRHS systems were defined in the context of algebraic cryptanalysis of symmetric ciphers [8], but have many different uses in cryptography and cryptanalysis (see survey paper [9]). For algebraic attacks, Raddum and Zajac have developed a specialized MRHS solver based on linear algebra and exhaustive search [10], which we now call the RZ solver. We provide more background on MRHS systems and the RZ solver in Sections II-B and II-C, respectively.

In this paper, we focus on the following research problem:

- Can the MRHS solver, specifically the RZ solver, be adapted to efficiently solve the syndrome decoding problem?

Our experiments indeed show that this is possible, although the MRHS solver is less efficient than the existing ISD methods. Still, the presented research can shed some light on the connection between the MRHS problem and the SDP, as well as provide new research directions for potential improvement of the presented methods.

Our main results are presented in three sections. In Section III we focus on the issue of representing the SDP via MRHS equation systems. We present two solutions. The explicit representation encodes the system along with weight constraint into a standalone MRHS system that can be solved by any MRHS solver. The implicit representation moves some work to the RZ solver and encodes just the SDP. The weight constraints must be checked by the modified solver during the search.

In Section IV we focus on experiments with random SDP instances of the hardest difficulty (code rate 0.5 and weight constraints set near Gilbert-Varshamov bound). The experiments aimed to assess different parameters of the MRHS representation (such as block size, and RHS order), and specific time-probability trade-off (based on the removal of some RHS vectors).

This research was supported in part by the NATO Science for Peace and Security Programme under Project G5985, and in part by the Slovak Scientific Grant Agency (Vedecká Grantová Agentúra MŠVVaŠ SR a SAV, Slovakia), Grant Number VEGA 1/0105/23.

Authors are with Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Bratislava, Slovakia (e-mail: pavol.zajac@stuba.sk, miloslav.smicik@gmail.com).

Finally, in Section V we present the complexity evaluation of the algorithm with the best settings and analyze also instances with different code rates.

II. Preliminaries

Notation: We will work with row vectors over binary field $\mathbb{F}_2$, denoted in bold type, such as $\mathbf{x} \in \mathbb{F}_2^n$. Matrices are also denoted in bold type, such as $\mathbf{M} \in \mathbb{F}_2^{n\times m}$. Hamming weight of vector $\mathbf{v}$ is denoted by $hw(\mathbf{v})$, and represents the number of non-zero elements of $\mathbf{v}$.

A. Syndrome decoding problem

Code-based cryptography is a branch of post-quantum cryptography that relies on the difficulty of problems related to coding theory. There are multiple definitions and variants of problems related to codes, but we will restrict our attention to the Syndrome Decoding Problem (SDP).

Let $\mathbf{s} \in \mathbb{F}_2^{n-k}$, and $\mathbf{H} \in \mathbb{F}_2^{n\times(n-k)}$, with $k < n$. Let $w \in \mathbb{Z}$, $0 \leq w \leq n$. Syndrome decoding problem (SDP) is a problem of finding $\mathbf{x} \in \mathbb{F}_2^n$, such that $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{s}$, and $hw(\mathbf{x}) = w$.

It is known that the decision version of the SDP (originally: coset weights problem, question of existence of $\mathbf{x}$ with $hw(\mathbf{x}) \leq w$) is an NP-complete problem [11]. In general, random instances of the problem (in certain parametric regimes) are believed to be difficult to solve. The most efficient class of algorithms for solving SDP are based on the so-called information set decoding (ISD) techniques [12]. An exact computational complexity of ISD algorithms depends on all parameters k, n, w, but in practice it can be approximated by 2^{cw} with $c = -\log_2(1 - k/n)$ (for small w) [13]. The syndrome decoding problem is also believed to be difficult to solve on quantum computers and is thus the basis of security of various post-quantum cryptographic systems, such as Classic McEliece [14], BIKE [13], and others.

In our research, we will focus on random linear binary codes. Given random $\mathbf{H} \in \mathbb{F}_2^{n\times(n-k)}$ as a parity check matrix, the corresponding linear binary code is a set of vectors $\mathcal{C} = \{\mathbf{c}; \mathbf{c}\mathbf{H}^T = 0\}$. It is known that with probability approaching 1 as $n \to \infty$, the minimum distance of a random binary linear code of length n and rate $R = k/n$ is at least $n\delta_{GV}(R)$ [15]. Here, $\delta_{GV}(R)$ is relative Gilbert-Varshamov distance, which is defined as the root $\delta < 1/2$ of the equation $H(\delta) = 1 - R$, where H is the binary entropy function, $H(p) = -p\log_2 p - (1 - p)\log_2(1 - p)$. Setting $R = 1/2$, and $w \approx n\delta_{GV}$ provides the hardest instances of the SDP (for random binary linear codes). The complexity of the SDP in this regime can then be estimated as $2^{\alpha n}$, with parameter α depending on the concrete algorithm. Table I from [16] summarizes complexity exponents of common ISD algorithms, as well as references for these algorithms.

B. Multiple Right-Hand Sides equation systems

The aim of this article is to explore alternative avenues for solving SDP. The idea is to convert instances of the SDP to a specific algebraic notation called a Multiple Right-Hand Sides (MRHS) equation system [8]. We then proceed to solve the MRHS system with a dedicated MRHS solver. In the following, we summarize the definitions of MRHS systems and provide the basic details of the RZ algorithm [10] that can be used to solve MRHS systems.

TABLE I
Asymptotic time complexity $O(2\alpha n)$ for major ISD algorithms in full-distance decoding setting [16]

Prange [17]	Dumer [18]	MMT [19]	BJMM [20]	MO15 [21]	BM18 [22]	Sieving ISD [23]
0.121	0.116	0.112	0.102	0.097	0.096	0.101

Let $\mathbf{M} \in \mathbb{F}_2^{n\times k}$, and let $S \subset \mathbb{F}_2^k$. A Multiple Right-Hand Sides equation (defined by the tuple $(\mathbf{M}, S)$) is a formal inclusion in the form

$$\mathbf{x}\mathbf{M} \in S.$$

Any vector $\mathbf{x} \in \mathbb{F}_2^n$, for which the inclusion holds, is called a solution of the MRHS equation.

A Multiple Right-Hand Sides equation system (MRHS system) is a set of MRHS equations defined by tuples $(\mathbf{M}_i, S_i)$ for $i = 1, 2, \ldots, m$ with common dimension n. Vector $\mathbf{x} \in \mathbb{F}_2^n$ is a solution of the MRHS system, if and only if it is a solution of each MRHS equation in the system.

An MRHS system can also be written in a joint form [10] by concatenating matrices on the left-hand side, and using a Cartesian product on the right-hand side:

$$\mathbf{x}\,(\mathbf{M}_1\mathbf{M}_2\cdots\mathbf{M}_m) \in S_1 \times S_2 \times \cdots \times S_m.$$

Note that the joint form of an MRHS system is an MRHS equation with (typically) a large right-hand side set that is not enumerated explicitly. We say that MRHS systems are polynomially bounded if they are members of a family of MRHS systems with the sizes of sets S_i as well as dimensions k_i and m restricted polynomially in n. There exists a polynomial time algorithm for verifying whether vector $\mathbf{x}$ is a solution of a polynomially bounded MRHS system. On the other hand, deciding whether some MRHS system has a solution is an NP-complete problem [24] and finding a solution of an MRHS system in general seems difficult.

MRHS systems have many applications in algebraic cryptanalysis [9], and potential use in post-quantum cryptography [25]. MRHS systems can be used as intermediate reductions between various types of NP-complete problems [6]. Specifically, after polynomial time reduction, we can use (regular) decoding algorithms to solve MRHS systems (a specific experimental version was provided in [26]). In this article we are interested in the opposite reduction: Is it possible to efficiently transform syndrome decoding problem instances into MRHS systems? How difficult is it to solve the resulting MRHS systems in practice with known MRHS-solving algorithms?

C. RZ solver

There are multiple methods proposed to solve MRHS systems. Original methods from [8] include so-called Agreeing and Gluing, which essentially rely on joining selected right-hand side sets, potentially removing some conflicting vectors in the process. Local reduction methods [27] generalize the

Agreeing approach, and combine guessing (a substitution of values of selected $\mathbf{x}$ components), and removing conflicts within right-hand side sets (similar to DPLL-based SAT solvers). Sparse MRHS systems can also be solved by heuristic methods related to bit-flipping and hill climbing [28].

In [10], Raddum and Zajac have proposed a systematic MRHS solver (we call it the RZ solver) that relies on linear algebra and fast exhaustive search. This MRHS solver has been actively developed and is available as an open-source tool at https://github.com/zajacpa/mrhs-solver.

The RZ solver starts with an MRHS system in the joint form. With linear algebra operations, the joint matrix is transformed to a specific reduced row echelon form. Suppose we have the following equation in block form:

$$(x_1,\ldots,x_k,\ldots,x_{k+p-1},\ldots,x_n)\cdot \left(\begin{array}{ccc|c} \mathbf{A} & \mathbf{0} & \mathbf{T} & \mathbf{B}_1 \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{B}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_3 \end{array}\right) \in S_1 \times \ldots \times S_l \times \cdots \times S_m. \quad (1)$$

Values up to x_{k-1} are known, and they satisfy the corresponding partial MRHS system based on the joint matrix $\mathbf{A}$ and right-hand side sets $S_1,\ldots,S_{l-1}$. The next MRHS equation (called a block) has p pivot columns corresponding to $\mathbf{I}$ part and some additional columns corresponding to $\mathbf{T}$, with the right-hand side set S_l. The rest of the system is given by joint matrix $\mathbf{B}$ and the remaining right-hand side sets.

The RZ solver tries to find values of p unknowns $x_k,\ldots,x_{k+p-1}$. Firstly, we need to compute $(x_1,\ldots,x_k)\cdot\mathbf{T}$, and restrict the set S_l to a subset of vectors compatible with this value. The first part of any of the remaining vectors can be substituted into $x_k,\ldots,x_{k+p-1}$. If there is no option, we mark the potential solution as incorrect and try to use backtracking to find another one. On the other hand, if there are one or more valid options, the algorithm uses depth-first search (with backtracking) through the potential solution space, until the whole vector $\mathbf{x}$ is constructed (or no solution is found). Additional speedups are available by efficient pre-computation and the use of look-up tables based on sets S_i.

The algorithm complexity is exponential in general, but it also depends on the exact form of the system (see analysis of random MRHS systems in [10]). In practice, the number of operations is influenced by the order of MRHS equations, as well as the order in which the right-hand sides are investigated during the depth-first search.

For the experiments with the decoding problem, we have created a modified version of the RZ solver. This version can keep track of the Hamming weight of the partial solution $(x_1,\ldots,x_{k-1})$. If the Hamming weight rises above the specified limit, we can mark the partial solution as incorrect. This operation can cut off some branches of the search tree that would be explored by an unmodified search procedure.

III. MRHS REPRESENTATION OF THE DECODING PROBLEM

Given a SDP instance $\mathbf{e}\cdot\mathbf{H}^T=\mathbf{s}$, $wh(\mathbf{x})\leq w$, we can transform it into a problem of finding a short vector in a specific code. Let $\mathbf{G}$ be a generator matrix of the binary linear code $\mathcal{C}$ with the parity check matrix $\mathbf{H}$. Let $\mathbf{u}$ be an arbitrary solution of $\mathbf{u}\cdot\mathbf{H}^T=\mathbf{s}$ (with any weight). If the SDP instance has a solution $\mathbf{e}$, the solution can be written as $\mathbf{e}=\mathbf{u}+\mathbf{c}$, where $\mathbf{c}=\mathbf{x}\mathbf{G}\in\mathcal{C}$. If w is lower than the minimum distance of the code $\mathcal{C}$, we can search for a low-weight codeword in a code generated by $\mathbf{G}'$, where $\mathbf{G}'$ is $\mathbf{G}$ with the added vector $\mathbf{u}$.

The most straightforward way of representing a problem of finding a low-weight codeword in a binary linear code generated by matrix $\mathbf{G}$ with the MRHS system is the following:

$$\mathbf{x}\mathbf{G}\in\{\mathbf{v};hw(\mathbf{v})\leq w\}. \quad (2)$$

Here $\mathbf{x}$ is an input vector, matrix $\mathbf{G}$ is a generator matrix of a given code, and the set $\{\mathbf{v};hw(\mathbf{v})<=w\}$ is the set of every possible vector with the required weight.

This representation, however, has a problem. The right-hand side set is too big for practical use even for smaller instances. An apparent solution in MRHS representation is to split the system into multiple parts. The system split can look like the following:

$$\mathbf{M}=\left(\begin{array}{cccc}\mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m\end{array}\right)$$

$$S=\prod_{i=1}^{m}\{\mathbf{x}\in\mathbb{F}_2^{n/m};hw(\mathbf{x})\leq w\}$$

where m is the total number of blocks. In the example, every block is of equal size, but this is not required. If the system is split into a sufficient amount of blocks, the right-hand side sets become small enough to be practical.

However, the example system does not properly take into account the constraint on the maximum weight of the vectors on the right-hand side. In the example with one block, we have constructed the RHS set directly to contain only vectors with the correct weight. When we split the vector into multiple blocks, we do not know the exact weight distribution between blocks. Thus we need to fill each set with all possible vectors (up to the maximal weight w, but this is typically higher than the dimension of the block). However, the combination of a sufficient number of partial vectors even of small weight can produce a joint vector that has a larger weight than is sought.

To address this problem, we have identified two practical ways of weight-constrained representation in MRHS systems. We call them implicit and explicit weight representation.

Implicit representation has a simple MRHS construction and relies on the solver to maintain a correct weight throughout the calculation. The RZ solver had to be modified in order to be able to calculate the weight, see Section II-C.

The second, so-called explicit representation, is a rather complicated MRHS construction that is capable of maintaining the correct weight within the MRHS system itself and can be solved with the original (unmodified) RZ solver, as well as any other type of MRHS solver. The representation relies on additional variables and equations that represent partial weights of current right-hand sides in a given block, and the propagation of these weights when joining individual RHS sets.

An MRHS system with explicit weight representation has the following block form of the joint matrix:

$$\mathbf{M} = \left(\begin{array}{ccc|ccc|ccc|c} 0 & \mathbf{G}_1 & 0 & 0 & \mathbf{G}_2 & 0 & 0 & \mathbf{G}_3 & 0 & \cdots \\ \mathbf{I}_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{I}_1 & \mathbf{I}_1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \mathbf{I}_2 & \mathbf{I}_2 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{I}_3 & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots \end{array}\right)$$

In the construction, each block contains initial weight part $\mathbf{I}_{i-1}$, part of the generator matrix $\mathbf{G}_i$, and the cumulative weight part $\mathbf{I}_i$.

The right-hand side sets are constructed as follows:

$$S_i = \bigcup_{c=0}^{w} \bigcup_{d=0}^{w-i} \bigcup_{l=0}^{\binom{|\mathbf{G}_i|}{d}} \{(\mathbf{c}, \mathbf{v}, \mathbf{e}); \\ \mathbf{c} = bin(c); \mathbf{v} = gen(|\mathbf{G}_i|, d, l); \mathbf{e} = bin(c+d).\} \tag{3}$$

The notation $bin(i)$ represents fixed size binary encoding of number i. The size of the encoded number should be $\lceil \log_2 w \rceil$ (the number of bits of w). Function $gen(i, j, k)$ creates the k-th vector of length i and weight j. Each tuple $(\mathbf{c}, \mathbf{v}, \mathbf{e})$ thus represents: initial weight c (of the partial solution before the i-th block), selected partial solution $\mathbf{v}$ (of the i-th block), and the cumulative weight $e = c + hw(\mathbf{v})$ (of the partial solution including the i-th block). In the initial S_1, we only include vectors with $c = 0$.

Note that the size of each S_i is at most $w^2 2^{l_i}$, where l_i is the dimension of $\mathbf{G}_i$. Typically, we select some fixed dimension l and then split the matrix into $m = n/l$ blocks (if l does not divide l, we create an extra block of smaller dimension). If we keep l fixed with growing m, we have created a polynomially bounded MRHS family (as $w \leq n$).

Note that for larger block dimensions l, and code weights w this system can still be quite large in practice. This is one of the reasons why we prefer to use implicit weight representation for the experiments. In the implicit representation, the system is stored in the form of equation (2). We still use the weight counters c, e, but they are only stored inside the RZ solver in specific data structures associated with the current partial solution.

IV. Experimental results: hard instances with fixed code rate 0.5

We have conducted several experiments to evaluate the proposed methods. The measurements were realized by our custom software tools. The whole technical solution consists of multiple parts and is described in the following subsection.

A. Tools and methods

The first step is the problem generation part. We have used a freely available syndrome decoding problem generator from the portal decodingchallenge.org [29], section Syndrome decoding problem. The software generates hard instances of the SDP based on random binary linear codes with fixed code rate 0.5, and w slightly higher than the Gilbert-Varshamov bound: $w = \lceil 1.05 n \delta_{GV} \rceil$. Using this generator we have generated 100 instances for every set of parameters. This was a one-time process.

The generated instances of decoding problems were then transformed into an MRHS instance with a custom Python script. This script is able to generate instances with explicit or implicit weight representation (parameter `solve_metohod`). The generated MRHS instances are also parameterized by the following parameters:

- `block_length` - Size of one block of the MRHS system.
- `rhs_reduction` - Reduction of RHS set. This parameter tries to eliminate less probable vectors within each RHS set.
- `rhs_order` - Ordering of RHS set. We can either sort the (remaining) vectors by their Hamming weights (ascending or descending) or shuffle them randomly.

We explore the effects of these parameters in the next subsection.

The generated MRHS instances are ready to be fed into the MRHS solver. We are using two versions of the MRHS solver. The first solver for explicit weight representation is the standard RZ solver. The second, modified solver, is the RZ solver with the ability to calculate the weight of the selected RHS during the calculation and to compute cumulative weight during the search process, as well as to backtrack the calculation process prematurely if the weight limit is reached. Note that the modified solver is based on the same source code. The weights are efficiently precomputed and stored within the data structures used by the RZ solver. The only extra operations within the main loop of the algorithm are the addition of previous weight and selected weight, and the weight comparison.

The transformation scripts and solvers are run by an automation script that handles the experiments. The script generates an instance with specified parameters, calls the solver, and stores the solver output into a corresponding result file. This process is repeated 100 times for each specific set of parameters. When not mentioned otherwise, we use the (logarithm of the) number of iterations of the main loop of the (modified) RZ algorithm (median value from 100 experiments) as our data points for the visualizations. Note that the size of the matrix also influences the total computation time, but this influence is linear, while the expected number of iterations grows exponentially. Thus, for small instances, the time statistic is too noisy to be useful, and for larger instances, the influence of the matrix size on computational time is negligible in comparison to the impact of the number of iterations.

B. Explicit vs. Implicit representation

Qualitative comparison between implicit and explicit weight representation is depicted in Figure 1. The experiments strongly suggest that there is a constant difference in complexity, with explicit representation requiring a smaller number of iterations of the RZ algorithm. For the sake of comparison, we also include an extra line `method_ww` that provides the number of iterations required to find a solution for the MRHS system without weight restrictions. This line represents

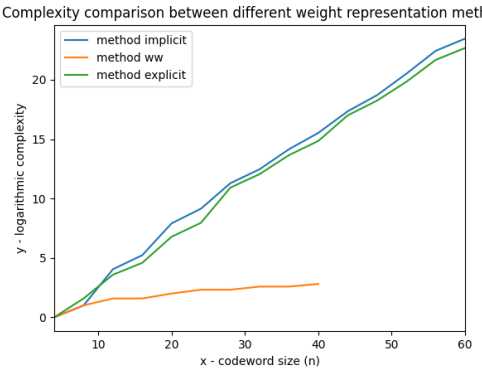

Fig. 1. Complexity comparison between different representation types

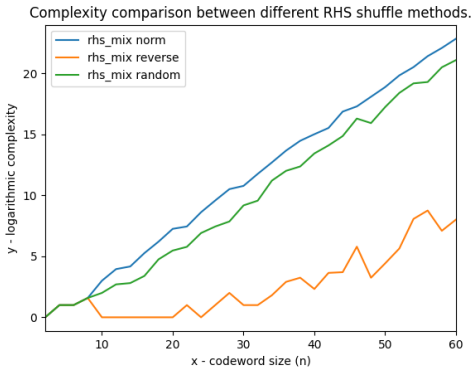

Fig. 2. Influence of the order of RHS elements on the complexity of the algorithm. Instances with block size 2 (implicit representation) are used.

TABLE II
MEDIAN ITERATIONS AND TIME OF 100 EXPERIMENTS, $n = 60$, RATIO OF EXPLICIT VS. IMPLICIT REPRESENTATION

	Implicit	Explicit	Ratio Ex/Im
Iterations [M]	11.5	6.7	58 %
Time [ms]	89.5	32.0	36 %

essentially a baseline complexity of solving the linear equation system.

The explanation of the difference between `method_implicit` and `method_explicit` lies in the processing of weight computation. In the explicit case, possible weight contributions are essentially precomputed in the representation. On the other hand, the implicit method requires that the solver explores additional branches before it can discard them. For example, if $w = 50$, and we have accumulated weight 49, explicit representation would only branch to weight-1 vectors in the next iteration, while implicit solver would explore vectors of every weight from the next RHS set.

Table II presents the concrete number of iterations and times obtained by implicit and explicit representation in the basic setting (without further optimizations discussed in later sections). The comparison of the total time of the explicit method is even more favorable than the comparison of the number of iterations. This further difference is caused by weight counters and their management in the modified RZ solver. In practical terms, however, the difference in the number of iterations and time is negligible between the implicit and explicit representations. In practice, if we include precomputation and preparation of the system, it is more efficient to work with the implicit representation (especially with RHS sets of low dimension). Thus, all other experiments use implicit representation with the modified RZ solver.

C. Influence of the RHS order

Our experiments have shown that the order of vectors in the RHS sets is significant to the algorithm's performance. The main results are shown in Figure 2. The most efficient order of RHS is `reverse`. This means that right-hand sides are ordered from the least probable to the most probable (probability depends on the Hamming weight of the RHS and the block size). However, the solver uses a LIFO queue to process the potential RHSs, thus processing the RHS set in this case from the most probable RHS to the least probable. ordered by probability in reverse order is most efficient. Thus, the results are as expected. What is less expected is that using the `norm` (the least probable RHS first), and `random` order was giving orders of magnitude worse results.

Note that RHS sets were in the same fixed order during the whole experiment instance. It might be interesting to also explore dynamic RHS selection (calculating conditional probabilities based on already accumulated weight), which we suspect would provide the best results. However, the current optimized architecture of the RZ solver implementation unfortunately does not support this kind of RHS processing.

D. Partial RHS sets

The currently best methods for solving the decoding problem are based on ISD methods. These methods are based on efficient algorithms that have a small chance of success. The complexity is then computed as the expected number of repetitions required to succeed with some specified probability. We can think of the setup of the ISD-based algorithm as a trade-off between computational probability and complexity.

In our base algorithm, we essentially search through the (potentially whole) space of low-weight vectors. As the previous experiments show, the selection of the path through the search tree is important to the expected complexity. We can however optimize the search further by cutting some branches, which seem to be less probable than the remaining ones. We again do the probability-complexity trade-off: there is a smaller space to search, but we have some probability that the algorithm fails to find a solution.

In practice, we reduce the search space by removing some vectors from RHS sets (each vector becomes one branch in the search tree). Firstly, we compute the probability that each RHS vector is a solution (based on Hamming weight) and then remove some fixed fraction of the RHS vectors (starting from the least probable ones and using random selection where probability is equal).

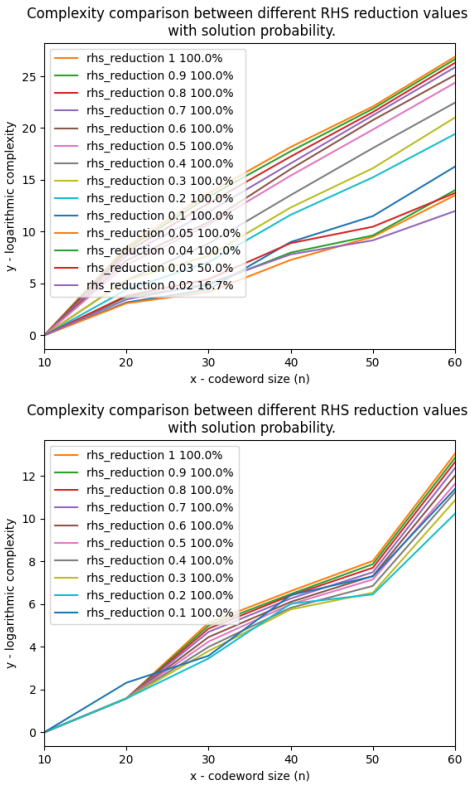

Fig. 3. Complexity (and success rate) depending on RHS reduction. The first number in the legend is the reduction rate and the second number is the success percentage. Data are obtained from instances with block size 10. On the left: `norm` order (the worst case). On the right: `reverse` order (the best case).

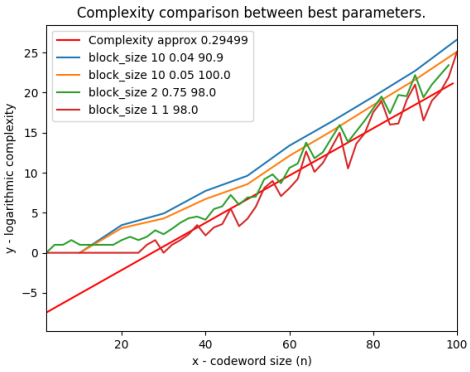

Fig. 4. Different combinations of parameters with an approximation

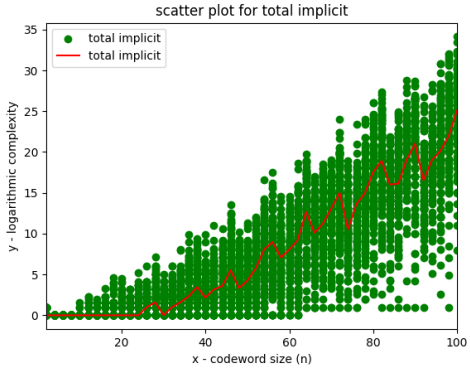

Fig. 5. Scatter graph of the best result

In Figure 3 we demonstrate the results of this method with blocksize 10 (each RHS has initially 2^{10} vectors). The experiments were fully successful until 4% of the original size (41 vectors in RHS, all vectors of weight 0, and 1, and most of the weight 2 vectors). Further decrease in RHS size caused the algorithm to fail in some cases.

The influence that the RHS reduction has is diminished if the RHS sets are explored in the correct order (from the most probable to the least probable), which is demonstrated by a difference in Figure 3. Even though the advantage of RHS reduction is lower in the case of correct RHS order, the best results are still obtained by combining both methods.

V. Overall Complexity Evaluation

In order to compare our MRHS representation of the syndrome decoding problem to current state-of-the-art decoding algorithms, we have combined the best sets of parameters and approximated their complexity coefficient α by least square metric. As a reminder, the asymptotic complexity estimate is $2^{\alpha n}$, with the parameter regime of random linear code and weight chosen according to the GV bound. Figure 4 depicts the results. The red straight line is the complexity approximation (numbers for small n are skewed due to trivial solutions). The calculated parameter $\alpha \approx 0.295$ is similar for different parametric settings of our algorithm (different block size, with optimal RHS order, and possible RHS reduction).

When using the implicit solver, the best results were achieved for block size 1 (no RHS reduction possible). Note that smaller block sizes lead to better performance (in terms of constants, not α), but with larger variance in results. Figure 5 contains a scatter plot for the best result and serves as an example of the dispersion of the results. Note that the y-axis is logarithmic, so the dispersion is quite large. The red line is the median of 100 runs (with the same size n). The main source of the dispersion is the fact that we stop the computation after the first valid vector, and the first vector can be anywhere in the solution space. This leads to essentially a log-normal distribution of results. In the graph, however, we can see that there are significant outliers and variances even in median values. This effect is suppressed for larger block sizes, where the selection of the partial vector contains more local information.

Our final experiment assessed the effect of code rate on the complexity of our algorithm, similar to the evaluation of ISD methods. Figure 6 summarizes an experiment with instances with different code rates (rather than the fixed code rate 0.5 used in previous experiments). Again, the codes were random, with weight expected near the GV bound. The size of the code was fixed to $n = 60$, block size 2 (no optimizations applied). The results are similar to the ISD algorithm, with the hardest instances localized around the code rate of 0.5 (maximum is

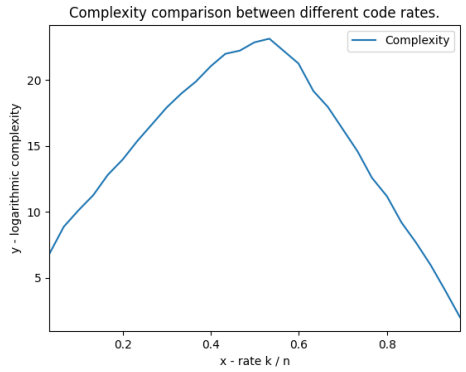

Fig. 6. Dependence of the computation complexity on the code rate

reached at $k/n = 0.533$), with a decreasing trend in both directions (both low and high code rates).

To conclude, our method is less efficient than the modern ISD algorithms but provides qualitatively similar results (with respect to code rate). To improve the complexity constant, we would need to improve the base algorithm used to solve the MRHS problem to include some of the time-memory trade-offs, or to be able to somehow exploit the collisions and birthday paradox. Another possibility is to use some more advanced probability-complexity trade-offs and advanced search algorithms. We believe that this study provides sufficient basis on which to study further algorithms and improvements of this method.

REFERENCES

[1] R. McEliece, "A public-key cryptosystem based on algebraic coding theory," Jet Propulsion Laboratory DSN Progress Report, Tech. Rep., 1978. [Online]. Available: http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF

[2] M. Repka and P. Zajac, "Overview of the McEliece cryptosystem and its security," *Tatra Mountains Mathematical Publications*, vol. 60, no. 1, pp. 57–83, 2014.

[3] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. Von Maurich, R. Misoczki, R. Niederhagen *et al.*, "Classic McEliece: conservative code-based cryptography," 2022, round 4 Submission.

[4] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the mceliece cryptosystem," in *Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA, October 17-19, 2008 Proceedings 2*. Springer, 2008, pp. 31–46.

[5] H. Randriambololona, "The syzygy distinguisher," Cryptology ePrint Archive, Paper 2024/1193, 2024. [Online]. Available: https://eprint.iacr.org/2024/1193

[6] P. Zajac, "Connecting the complexity of MQ-and code-based cryptosystems," *Tatra Mountains Mathematical Publications*, vol. 70, no. 1, pp. 163–177, 2017. [Online]. Available: https://doi.org/10.1515/tmmp-2017-0025

[7] ——, "Polynomial reduction from syndrome decoding problem to regular decoding problem," manuscript.

[8] H. Raddum and I. Semaev, "Solving multiple right hand sides linear equations," *Designs, Codes and Cryptography*, vol. 49, pp. 147–160, 2008.

[9] P. Zajac, "Algebraic cryptanalysis with MRHS equations," *Cryptography*, vol. 7, no. 2, p. 19, 2023.

[10] H. Raddum and P. Zajac, "MRHS solver based on linear algebra and exhaustive search," *Journal of Mathematical Cryptology*, vol. 12, no. 3, pp. 143–157, 2018. [Online]. Available: https://doi.org/10.1515/jmc-2017-0005

[11] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978. [Online]. Available: https://doi.org/10.1109/TIT.1978.1055873

[12] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "A finite regime analysis of information set decoding algorithms," *Algorithms*, vol. 12, no. 10, 2019. [Online]. Available: https://www.mdpi.com/1999-4893/12/10/209

[13] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Ghosh, S. Gueron, T. Güneysu *et al.*, "BIKE: Bit Flipping Key Encapsulation," INRIA, Tech. Rep., 2022. [Online]. Available: https://inria.hal.science/hal-04278509

[14] D. Bernstein, T. Chou, T. Lange, I. von Maurich, R. Misoczki, R. Niederhagen, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, and W. Wang, "Classic mceliece," 2017. [Online]. Available: https://research.tue.nl/en/publications/classic-mceliece

[15] A. Barg and G. D. Forney, "Random codes: Minimum distances and error exponents," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2568–2573, 2002.

[16] S. Narisada, S. Uemura, H. Okada, H. Furue, Y. Aikawa, and K. Fukushima, "Solving McEliece-1409 in one day — cryptanalysis with the improved BJMM algorithm," Cryptology ePrint Archive, Paper 2024/393, 2024, https://eprint.iacr.org/2024/393. [Online]. Available: https://eprint.iacr.org/2024/393

[17] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[18] I. Dumer, "On minimum distance decoding of linear codes," in *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*. Moscow, 1991, pp. 50–52.

[19] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $o(2^{0.054n})$," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 107–124.

[20] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in $2^{n/20}$: How 1+ 1= 0 improves information set decoding," in *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*. Springer, 2012, pp. 520–536.

[21] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 203–228.

[22] L. Both and A. May, "Decoding linear codes with high error rate and its impact for lpn security," in *International Conference on Post-Quantum Cryptography*. Springer, 2018, pp. 25–46.

[23] Q. Guo, T. Johansson, and V. Nguyen, "A new sieving-style information-set decoding algorithm," Cryptology ePrint Archive, Paper 2023/247, 2023, https://eprint.iacr.org/2023/247. [Online]. Available: https://eprint.iacr.org/2023/247

[24] P. Zajac, "MRHS equation systems that can be solved in polynomial time," *Tatra Mountains Mathematical Publications*, vol. 67, no. 1, pp. 205–219, 2016. [Online]. Available: https://doi.org/10.1515/tmmp-2016-0040

[25] P. Zajac and P. Spacek, "A new type of signature scheme derived from a MRHS representation of a symmetric cipher," *INFOCOMMUNICATIONS JOURNAL*, vol. 11, no. 4, pp. 23–30, 2019.

[26] P. Zajac, "Upper bounds on the complexity of algebraic cryptanalysis of ciphers with a low multiplicative complexity," *Designs, Codes and Cryptography*, vol. 82, pp. 43–56, 2017.

[27] ——, "Using local reduction for the experimental evaluation of the cipher security," *Computing and Informatics*, vol. 37, no. 2, pp. 349–366, 2018.

[28] ——, "On solving sparse MRHS equations with bit-flipping," *Publ.Math.Debrecen*, vol. 100 / Supplementum, pp. 683–700, 2022.

[29] M. L. Julien Lavauzelle, Matthieu Lequesne, "Challenges for code-based problems," 2023. [Online]. Available: https://decodingchallenge.org/