An improvement to Stern's algorithm

Johansson, Thomas; Löndahl, Carl

2011

An improvement to Stern's algorithm*

Thomas Johansson and Carl Löndahl

Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{thomas, carl}@eit.lth.se

Abstract. The decoding problem is a fundamental problem in computational complexity theory. In particular, the efficiency of which the problem can be decided has implications on the security of cryptosystems based on hard problems in coding theory. Stern's algorithm has long been the best algorithm available, with slight modifications over the years yielding only small speed-ups. This paper describes an improved method of finding low weight codewords in a random code, leading to an improved decoding algorithm.

Keywords: information set decoding, random codes, public-key cryptography, birthday technique, binary codes.

1 Introduction

Error-correcting codes are wide-spread in the realm of information theory and have found applications in several areas of cryptography. Due to theoretical advances in quantum computation, which renders many classical cryptosystems obsolete, cryptography based on hard problems in coding theory has become more interesting.

A linear $[n, k]$-code is a linear subspace of $\mathbb{F}_q^n$, defined as $\mathcal{C} = \left\{\mathbf{uG} : \mathbf{u} \in \mathbb{F}_q^k\right\}$, where $\mathbf{uG} \in \mathbb{F}_q^n$. The matrix $\mathbf{G}$ is called the *generator matrix* and corresponds to a linear map $\mathbb{F}_q^k \to \mathbb{F}_q^n$, where $\mathbf{u} \mapsto \mathbf{v}$. In this paper, we will only consider the binary case, i.e. $q = 2$. The code $\mathcal{C}$ can also be defined by its *parity check matrix* $\mathbf{H}$, as $\mathcal{C} = \left\{\mathbf{v} \in \mathbb{F}_q^n : \mathbf{Hv}^T = \mathbf{0}\right\}$.

The elements of the code $\mathcal{C}$ are called codewords and the distance between two codewords, $\mathbf{x}, \mathbf{y}$, is measured through the *Hamming distance*, denoted $d_H(\mathbf{x}, \mathbf{y})$. This is defined as the number of positions in which $\mathbf{x}$ and $\mathbf{y}$ differ. Another important measure is the *Hamming weight*, denoted $w_H(\mathbf{x})$, which is the number of non-zero entries in the vector $\mathbf{x}$. Hamming distance and Hamming weight are connected by the identity $w_H(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0})$. The minimum distance d of the code $\mathcal{C}$ is the smallest distance among all pairs of disjoint codewords in $\mathcal{C}$. The minimum distance of a linear $\mathcal{C}$ can be computed as $\min_{\mathbf{v} \in \mathcal{C} \setminus \mathbf{0}} w_H(\mathbf{v})$. The error-correcting capability $t = \lfloor (d-1)/2 \rfloor$ of a code $\mathcal{C}$ is equal to the maximum number of errors it can correct.

* Just before finalizing this paper we found that similar work had been done in parallel by May, Meurer and Tomae [8] and their work will appear at Asiacrypt 2011.

The Nearest Codeword Problem (NCP) is defined as follows: given a linear code $\mathcal{C}$, a received message vector $\mathbf{r} = \mathbf{v} + \mathbf{e}$, where $\mathbf{v}$ is an unkown codeword in $\mathcal{C}$ and $\mathbf{e}$ is an unkown error vector of weight less than or equal to t, find the closest codeword $\mathbf{v} \in \mathcal{C}$. The assumption here is that there exists a unique closest codeword. NCP is shown to be $\mathcal{NP}$-complete (by reduction from for instance the well-known Max-Cut Problem) and this difficulty has been used in public-key cryptosystems such as McEliece PKC [9] and identification schemes such as the one by Stern [12].

The basic idea of basing security upon the hardness of NCP is this: an attacker without further knowledge will face the challenge of decoding a seemingly random code. As the code will have no obvious structure to exploit, the attacker cannot use efficient decoding algorithms. It is important to point out that since the codes used in coding-based cryptography are not truly random codes, it is not necessarily true that decoding that particular subset of codes used in the construction is hard. Thus, there may exist subexponential or even polynomial-time decoding algorithms. Such attacks are often referred to as *structural attacks*. Codes which open up for structural attacks should be refuted in the context of public-key cryptography. An example of this is the use of Reed-Solomon codes as underlying codes in McEliece PKC, as noted by Sidelnikov and Shestakov in [11].

In this paper, we will assume that the code actually is truly random. Since the problem is in $\mathcal{NP}$, we cannot in the worst case do better than exponential complexity assuming $\mathcal{P} \neq \mathcal{NP}$. Still, it is important to study how efficient the decoding can be done. In particular, this gives guidelines when building coding-based cryptography. The naive approach is to perform brute-force search over all $\binom{n}{w}$ possible weight w vectors. It is however possible to do more efficient than this. These attacks are often referred to as *generic decoding algorithms*.

The best known algorithms rely on *information set decoding* (ISD), which was introduced in a paper by Prange [10]. The idea behind information set decoding is to pick a sufficiently large set of error-free positions in a sent codeword such that the corresponding columns in the generator matrix form an invertible submatrix. The information sequence then can be obtained by multiplication of the corresponding subvector of the received vector and the inverted submatrix. Over the years, this simple technique has undergone further improvements in papers by Lee and Brickell [6], Leon [7] and Stern [5].

The ISD algorithms solve a problem which basically is the same as NCP, namely the function problem version of Subspace Weights. The function problem is to find a codeword of a certain weight w in a given code $\mathcal{C}$. In code-based cryptosystems, the number of errors e are generally kept constant and it is no more than half of the minimum distance of the underlying code $\mathcal{C}$. This means that if one extends the code $\mathcal{C}$ with the received sequence $\mathbf{r}$, i.e. if we can find a codeword of weight e in the code $\mathcal{C}'$ generated by $\mathbf{G}' = \left[\mathbf{G}^T \ \mathbf{r}^T\right]^T$, then we know the error sequence $\mathbf{e}$ since $\mathbf{e} \in \mathcal{C}'$. Given $\mathbf{e}$, one can easily obtain the information.

Our new algorithm differs from Stern's algorithm in that we choose the initial pairs from a single set of systematic positions instead two disjoint ones, intro-

ducing more degrees of freedom. Instead of a single list that is sorted and where elements then are matched together, we perform the procedure twice. Instead of assuming one error-free field, we assume that two fields are error-free. These ideas are similiar to ideas used in [4] for solving the KNAPSACK PROBLEM.

The paper is organized as follows. In Section 2, we present some classical algorithms and in Section 3, we present our new algorithm and explain the ideas behind it. We give complexity analysis and claims in Section 4. Section 5 concludes the paper.

2 Previous work

As mentioned before, SUBSPACE WEIGHTS is the problem of determining if there exists a codeword of weight w in a random linear $[n, k]$-code $\mathcal{C}$, assuming one such codeword exists. Since SUBSPACE WEIGHTS is $\mathcal{NP}$-complete, the problem solved by the ISD algorithms is $\mathcal{NP}$-hard.

There exist many algorithms for solving the function problem version of SUBSPACE WEIGHTS. We go through the most significant ones in chronological order.

Lee-Brickell algorithm The Lee-Brickell algorithm is a probabilistic algorithm intially proposed in [6]. It is the simplest non-trivial decoding algorithm and it serves as basis for other improvements. The Lee-Brickell algorithm works as follows: In practice, the first step is generally performed by choosing the columns

Algorithm 1 Lee-Brickell's algorithm

Input: Generator matrix $\mathbf{G}$ with k rows and n columns, parameters p

1. Choose a column permutation and form $\pi(\mathbf{G})$, where π is a random column permutation and $\mathbf{G}$ is the given generator matrix.
2. Bring the generator matrix $\pi(\mathbf{G})$ to systematic form:
$$\mathbf{G}' = \left[\mathbf{I}\ \mathbf{J}\right],$$
where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{J}$ is a $k \times n - k$ matrix.
3. Let $\mathbf{u}$ run through all weight p vector of length k. For each $\mathbf{u}$, check if the weight of $\mathbf{uG}$ is $w - p$. If no such codeword is found, return to 1.

one at the time and checking whether the chosen one is linearly independent of the other.

The parameter p is optimized given n and k to assert the lowest possible computational complexity.

Leon's algorithm This algorithm was proposed by Leon in [7]. It differs from Lee-Brickell's algorithm by assuming a field of bits of size i to be error-free, i.e. the sum of the rows of $\mathbf{G}$ in a given interval is assumed to have weight 0.

Algorithm 2 Leon's algorithm

Input: Generator matrix $\mathbf{G}$ with k rows and n columns, parameters p, l

1. Choose a column permutation and form $\pi(\mathbf{G})$, where π is a random column permutation and $\mathbf{G}$ is the given generator matrix.
2. Bring the generator matrix $\pi(\mathbf{G})$ to systematic form:

$$\mathbf{G}' = \left[\mathbf{I}\ \mathbf{L}\ \mathbf{J}\right],$$

   where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n-k-l$ matrix.
3. Let $\mathbf{u}$ run through all weight at most p vector of length k. If $\mathbf{x} = \mathbf{u}\left[\mathbf{I}\ \mathbf{L}\right]$ has weight less than p, compute the whole codeword.

The largest complexity cost of the algorithm is in the first (Gaussian) step. It was later modified by Canteaut and Chabaud in [2] to reduce the complexity of the first step. In their modification, the Gaussian step is performed once and then for each iteration a column from the information set and a column from the redudant set are swapped. Hence, the expensive Gaussian step can be omitted. The authors show that the algorithm still converges using this method.

Stern's algorithm Proposed by Stern in [5], this algorithm further improves the algorithm by Leon using a birthday technique. The previously mentioned algorithms and their improvements all have the same running time up to a polynomial factor. Stern's algorithm, however, slightly improves the decoding exponent.

Definition 1. *Let the set*

$$W_p = \{\mathbf{x} \in \mathbb{F}_2^n : |\{i : x_i \neq 0, 1 \leq i \leq k\}| = p\}$$

be the set of vectors that have weight p in the first k positions.

Stern's algorithm described using the generator matrix is as follows. The algorithm works with generator matrices $\mathbf{G}'$ in the following systematic form:

$$\mathbf{G}' = \left[\mathbf{I}\ \mathbf{L}\ \mathbf{J}\right],$$

where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n-k-l$ matrix. Let $\phi^{\mathbf{L}}(\mathbf{x}) \in \mathbb{F}_2^l$ be the value of $\mathbf{x}$ in positions in $\mathbf{L}$, i.e.

$$\phi^{\mathbf{L}}(\mathbf{x}) = \left[x_{k+1}\ x_{k+2} \cdots x_{k+l}\right].$$

Algorithm 3 Stern's algorithm

Input: Generator matrix $\mathbf{G}$ with k rows and n columns, parameters p, l

1. Choose a column permutation and form $\pi(\mathbf{G})$, where π is a random column permutation and $\mathbf{G}$ is the given generator matrix.
2. Bring the generator matrix $\pi(\mathbf{G})$ to systematic form:

$$\mathbf{G}' = \left[\mathbf{I}\ \mathbf{L}\ \mathbf{J}\right],$$

   where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n-k-l$ matrix.
3. Let $\mathbf{u}$ run through all weight p vectors of length $k/2$. Store all vectors $\mathbf{x} = (\mathbf{u}, \mathbf{0})\mathbf{G}'$ in a sorted list H_0, sorted according to $\phi^{\mathbf{L}}(\mathbf{x})$. Then construct a list H_1 sorted according to $\phi^{\mathbf{L}}(\mathbf{x})$, containing all vectors $\mathbf{x} = (\mathbf{0}, \mathbf{u})\mathbf{G}'$ where $\mathbf{u}$ runs through all weight p vectors. Add all pairs of vectors $\mathbf{x} \in H_0$ and $\mathbf{x}' \in H_1$ for which $\phi^{\mathbf{L}}(\mathbf{x}) = \phi^{\mathbf{L}}(\mathbf{x}')$ and put in a list H_2.
4. For each $\mathbf{x} \in H_2$, check if the weight of $\mathbf{x}$ is $w - 2p$. If no such codeword is found, return to 1.

Finiasz-Sendrier algorithm This algorithm, suggested in [3], solves NCP using the parity check matrix $\mathbf{H}$. This problem sometimes referred to as Computational Syndrome Decoding. The algorithm operates basically in same manner as Stern's algorithm, using the birthday trick. Given a binary matrix $\mathbf{H}$, a binary vector $\mathbf{s}$ and a non-negative integer w, find a binary vector $\mathbf{e}$ such that $\mathbf{e}\mathbf{H}^T = \mathbf{s}$. The parity check matrix $\mathbf{H}$ is divided as follows:

$$\mathbf{H} = \left[\mathbf{L}\ \mathbf{J}\right],$$

where $\mathbf{L}$ is an $n \times l$ matrix and J an $n \times k - l$ matrix. The steps of the algorithm are:

Algorithm 4 Finiasz-Sendrier algorithm

Input: Parity check matrix $\mathbf{H}$ with n rows and k columns, parameter l

1. Choose a column permutation and form $\mathbf{H}' = \pi(\mathbf{H})$, where π is a random column permutation and $\mathbf{H}$ is the given parity check matrix.
2. Let $\mathbf{e}_1$ run through all weight $\lfloor w/2 \rfloor$ vectors of length n and store it in a list H indexed by $\phi^{\mathbf{L}}(\mathbf{e}_1\mathbf{H}^T)$.
3. Let $\mathbf{e}_2$ run through all weight $\lceil w/2 \rceil$ and calculate $\mathbf{x} = \phi^{\mathbf{L}}(\mathbf{e}_2\mathbf{H}^T)$. For each such $\mathbf{x}$, go through all elements stored at position $\mathbf{x}$ in H and check whether $\mathbf{e}_1\mathbf{H}^T = s \oplus \mathbf{e}_2\mathbf{H}^T$. If so, return $\pi^{-1}(\mathbf{e}_1 \oplus \mathbf{e}_2)$ else continue.

Ball-collision decoding Contrary to assuming fields of error-free bits, it was proposed by Bernstein, Lange and Peters in [1] to allow $q > 0$ errors. We will not go into the description of ball-collision decoding, as it is not related to our approach. However, we mention that it exists and that it possibly might be combined with the approach we present, yielding further improvement in complexity. We will also use the numerical complexities given in [1] to compare with.

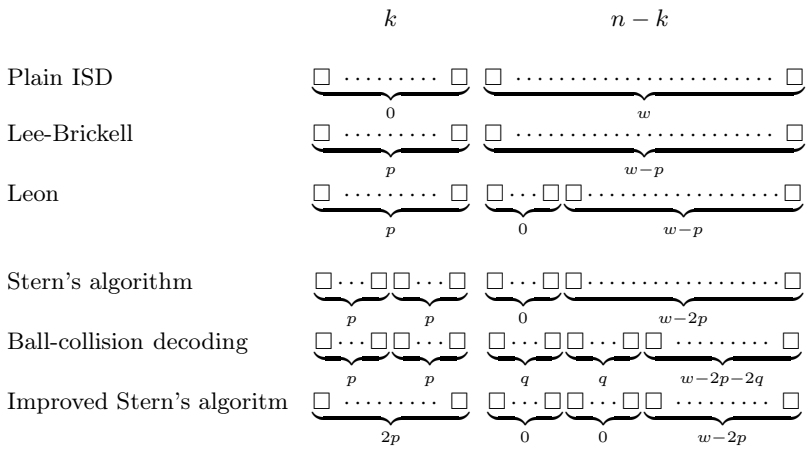

Fig. 1. Assumed error distributions for decoding algorithms.

3 A new improved algorithm

The algorithm will be working with generator matrices $\mathbf{G}'$ in the following systematic form:

$$\mathbf{G}' = \left[\mathbf{I}\ \mathbf{Z}\ \mathbf{L}\ \mathbf{J}\right],$$

where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{Z}$ is a $k \times z$ matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n - k - z - l$ matrix. Let $\phi^{\mathbf{Z}}(\mathbf{x}) \in \mathbb{F}_2^z$ be the value of $\mathbf{x}$ in positions in $\mathbf{Z}$, i.e.

$$\phi^{\mathbf{Z}}(\mathbf{x}) = \left[x_{k+1}\ x_{k+2} \cdots x_{k+z}\right].$$

Similarly, let $\phi^{\mathbf{L}}(\mathbf{x}) \in \mathbb{F}_2^l$ be the value of $\mathbf{x}$ in positions in $\mathbf{L}$, i.e.

$$\phi^{\mathbf{L}}(\mathbf{x}) = \left[x_{k+z+1}\ x_{k+z+2} \cdots x_{k+z+l}\right].$$

3.1 Explaining the ideas behind the algorithm

First we note that the algorithm is similar to Stern's algorithm in its steps. It makes a number of iterations, where each iteration can be successful, i.e. finding the weight w codeword, with a small probability.

Algorithm 5 Improved Stern

Input: Generator matrix $\mathbf{G}$, parameters p, z, l

1. Let $\mathbf{G}' = \pi(\mathbf{G})$, where π is a random column permutation and $\mathbf{G}$ is the generator matrix with k rows and n columns.
2. Bring the generator matrix $\mathbf{G}'$ to systematic form:

$$\left[\mathbf{I}\ \mathbf{Z}\ \mathbf{L}\ \mathbf{J}\right],$$

   where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{Z}$ is a $k \times z$ matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n - z - l$ matrix.
3. Let $\mathbf{u}$ run through all weight p vector of length k. Store all vectors $\mathbf{x} = \mathbf{u}\mathbf{G}'$ such that $\phi^{\mathbf{Z}}(\mathbf{x}) = \left[0\ 0 \cdots 0\right]$ in a sorted list H_1, sorted according to $\phi^{\mathbf{L}}(\mathbf{x})$. This is done by constructing a list H_0 containing all vectors $\mathbf{x} = \mathbf{u}\mathbf{G}'$ where $\mathbf{u}$ runs through all weight $p/2$ vectors. Then add all pairs of vectors $\mathbf{x}, \mathbf{x}' \in H_0$ in the list with $\phi^{\mathbf{Z}}(\mathbf{x}) = \phi^{\mathbf{Z}}(\mathbf{x}')$ and such that the largest index of the nonzero entries in $\mathbf{x}$ is smaller than the smallest index of nonzero entries in $\mathbf{x}'$.
4. As above, combine the list H_1 with itself to receive a new list H_2 of all codewords $\mathbf{x} = \mathbf{u}\mathbf{G}'$ with $\mathbf{u}$ of weight $2p$, such that $\phi^{\mathbf{L}}(\mathbf{x}) = \left[0\ 0 \cdots 0\right]$.
5. For each $\mathbf{x} \in H_2$, check if weight of $\mathbf{x}$ is $w - 2p$. If no such codeword is found, return to 1.

Each iteration starts with selecting a random permutation of the columns in the generator matrix, and bringing it to systematic form. As for Stern, this means that the nonzero entries in the low weight codeword appear in randomly selected positions in the codeword.

In an iteration, we require that in the first k positions (systematic part) of the desired codeword has weight $2p$, and in the following $z + l$ positions has weight 0. We can hope for a successful iteration, if this condition is fulfilled. Stern's algorithm has a similar (but not the same) condition.

The main new idea of this paper comes by introducing a second condition, that needs to hold for an iteration to be successful. A second necessary condition lowers the probability of a successful iteration, which is not to our benefit. However, it also lowers the number of active candidates in each iteration, hence reducing the computational complexity of each iteration.

The second condition can be explained as follows. Assuming that the first condition is valid, our desired codeword $\mathbf{c}_w$ of weight w has weight $2p$ in the first k positions, and in the following $z + l$ positions has weight 0. Then we can see that we will successfully find $\mathbf{c}_w$ in the iteration if there are two words $\mathbf{x}, \mathbf{x}' \in W_p$ such that $\mathbf{c}_w = \mathbf{x} + \mathbf{x}'$, and $\phi^{\mathbf{Z}}(\mathbf{x}) = \left[0\ 0 \cdots 0\right]$.

Since $\phi^{\mathbf{Z}}(\mathbf{c}_w) = \left[0\ 0 \cdots 0\right]$ from our first condition, we have $\phi^{\mathbf{Z}}(\mathbf{x}') = \left[0\ 0 \cdots 0\right]$, so if $\mathbf{x}$ remains in the list H_1, then so must $\mathbf{x}'$.

We write this condition as

$$\exists \mathbf{x}, \mathbf{x}' \in W_p \ :\ \phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}. \tag{1}$$

An overview picture of the reductions of the different lists done in the algorithm is given in Figure 2.

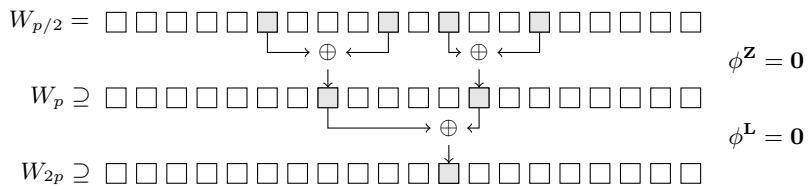

Fig. 2. An overview picture of how the lists are processed in the algorithm.

4 Complexity analysis

We derive formulas for the computational complexity similar to existing complexity formulas for the Stern algorithm. Note that we consider the basic approach where some implementation-oriented improvements have not been made.

The proposed algorithm has a structure similar to Stern's algorithm and we can use the same approach to evaluate its computational complexity. The algorithm runs a number of iterations and a first requirement is that the error vector after the random permutation is such that there are $2p$ errors in the first k positions, followed by zero errors in the $z + l$ following bits. The probability for this event is denoted $\mathcal{W}$,

$$\mathcal{W} = \mathbf{Pr}\left[w_H\left(\phi^{\mathbf{I}}(\mathbf{e})\right) = 2p, w_H\left(\phi^{\mathbf{Z}}(\mathbf{e})\right) = 0, w_H\left(\phi^{\mathbf{L}}(\mathbf{e})\right) = 0\right],$$

where e is a weight w error vector of length n. This probability is

$$\mathcal{W} = \frac{\binom{k}{2p}\binom{n-k-z-l}{w-2p}}{\binom{n}{w}}. \tag{2}$$

1. In the first step of an iteration, we apply a random permutation and transform the generator to systematic form. The complexity for this step, denoted A_0, is the same as for Stern's algorithm and is as in previous work set to

$$A_0 = (n-k)^2(n+k)/2. \tag{3}$$

2. Next, we construct all codewords with weight $p/2$ in the information set part. They are put in an array sorted according to $\phi^{\mathbf{Z}}(\mathbf{x})$. The complexity of this step, denoted A_1 is

$$A_1 = \binom{k}{p/2} c_1, \tag{4}$$

where c_1 denotes the actual work that has to be done to find $\phi^{\mathbf{Z}}(\mathbf{x})$ and then possibly storing more. The standard approach would set $c_1 = pz/2$ as we need to sum $p/2$ vectors over z bits to find out $\phi^{\mathbf{Z}}(\mathbf{x})$.

3. We now come to the filtering step. We build a new array of all codewords with weight p on the information set such that $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$, by adding pairwise all weight $p/2$ vectors with the same $\phi^{\mathbf{Z}}(\mathbf{x})$. Assume that we only add weight $p/2$ vectors that result in weight p in the first k bits. As every pairwise addition then gives a weight p vector such that $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$ and we assume that only a fraction 2^{-z} of all weight p vector will have $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$.

Observation 1 *There exist several pairs of vectors of weight $p/2$ that sum up to the same weight p vector.*

In fact, we will have $\binom{p}{p/2}$ pairs that sum up to the same vector. Since we desire to pick only one among these, we add a condition which is true for one and only one pair. Let $\mathcal{I}_1 = \{i_1, \ldots, i_{p/2}\}$ and $\mathcal{I}_2 = \{j_1, \ldots, j_{p/2}\}$ be (disjoint) sets of indices. Now assume that there exists a codeword with indices $\mathcal{I}_1 \cup \mathcal{I}_2$ which fulfills the condition $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$. As a consequence, the condition $\max \mathcal{I}_1 < \min \mathcal{I}_2$ will be true for one and only one pair of indices, since we generated all possible weight $p/2$ vectors without restriction. The filtering step can only be applied once with non-zero probability of loss of correct codewords. We show this by assuming the contrary and finding a counter-example. This is given in Example 1.

Example 1 *Assume that $p = 4$, that $\mathbf{a}_i, i \in [1, 8]$ are rows of $\mathbf{G}$ and $\phi^{\mathbf{Z}}(\mathbf{a}_1 \oplus \cdots \oplus \mathbf{a}_4) = \phi^{\mathbf{Z}}(\mathbf{a}_5 \oplus \cdots \oplus \mathbf{a}_8)$ and $\phi^{\mathbf{J}}(\mathbf{a_1} \oplus \cdots \oplus \mathbf{a}_4) = \phi^{\mathbf{J}}(\mathbf{a}_5 \oplus \cdots \oplus \mathbf{a}_8)$. This implies one (or all) of three cases:*

$$\phi^{\mathbf{Z}}(\mathbf{a_1} \oplus \mathbf{a_2}) = \begin{cases} \phi^{\mathbf{Z}}(\mathbf{a}_3 \oplus \mathbf{a}_4) \\ \phi^{\mathbf{Z}}(\mathbf{a}_5 \oplus \mathbf{a}_6) \\ \phi^{\mathbf{Z}}(\mathbf{a}_7 \oplus \mathbf{a}_8) \end{cases}$$

Now, assume that the only true case is $\phi^{\mathbf{Z}}(\mathbf{a}_1 \oplus \mathbf{a}_2) = \phi^{\mathbf{Z}}(\mathbf{a}_7 \oplus \mathbf{a}_8)$. This implies that $\phi^{\mathbf{Z}}(\mathbf{a}_3 \oplus \mathbf{a}_4) = \phi^{\mathbf{Z}}(\mathbf{a}_5 \oplus \mathbf{a}_6)$ and that $\phi^{\mathbf{Z}}(\mathbf{a}_1 \oplus \mathbf{a}_2) \neq \phi^{\mathbf{Z}}(\mathbf{a}_5 \oplus \mathbf{a}_6)$. The use of filtering now implies that the two concatenated codewords are $\mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \mathbf{a}_7 \oplus \mathbf{a}_8$ and $\mathbf{a}_3 \oplus \mathbf{a}_4 \oplus \mathbf{a}_5 \oplus \mathbf{a}_6$, and that these are the only codewords involving these indices. The conclusion is that repeated use of filtering would yield that the codewords could not be concatenated since $8 > 3$ and $6 > 1$.

This filtering step gives rise to complexity

$$A_2 = \binom{k}{p} 2^{-z} c_2, \tag{5}$$

where c_2 is the cost of computing the value of $\phi^{\mathbf{L}}(\mathbf{x})$. The standard approach would set $c_2 = pl$ as we need to sum p vectors over l bits to find out $\phi^{\mathbf{L}}(\mathbf{x})$.

4. The final step is the generation of a subset of all weight $2p$ vectors $\mathbf{x}$ such that $\phi^{\mathbf{L}}(\mathbf{x}) = \mathbf{0}$. The array contains about $\binom{k}{p}2^{-z}$ elements, and we sum together any two of them, $\mathbf{x}, \mathbf{x}'$, with the same $\phi^{\mathbf{L}}(\mathbf{x}) = \phi^{\mathbf{L}}(\mathbf{x}')$. As is argued in the analysis of Stern's algorithm, the total number of weight $2p$ vectors $\mathbf{x}'' = \mathbf{x} + \mathbf{x}'$ such that $\phi^{\mathbf{L}}(\mathbf{x}'') = \mathbf{0}$ is then $\left(\binom{k}{p}2^{-z}\right)^2 2^{-l}$. For each of them we need to check if its weight is w and we get the complexity for this part as
$$A_3 = \left(\binom{k}{p}2^{-z}\right)^2 2^{-l} c_3,$$
where c_3 is the cost of deciding if a vector is the desired vector. Following previous work, the standard value for c_3 would be $c_3 = 2p(n-k)$.

The final issue is the following. Assuming that the desired low weight codeword has $2p$ ones in the first k positions, then zero ones in the next $z+l$ positions, what is the probability that this codeword is actually in the subset of the remaining codewords of weight $2p$ in the information set. Well, in the process of building this subset there is one filtering stage. Namely, we keep only vectors with p ones in the first k positions such that $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$.

Note that if the desired codeword $\mathbf{x}''$ can be written as $\mathbf{x}'' = \mathbf{x} + \mathbf{x}'$ where $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$ then we must also have $\phi^{\mathbf{Z}}(\mathbf{x}') = \mathbf{0}$. This means that the desired codeword $\mathbf{x}''$ remains in the subset if there is an x such that $\phi^{\mathbf{Z}}(\mathbf{x}) = \mathbf{0}$.

As $\mathbf{x}''$ has $2p$ ones in first k positions, there are $\binom{2p}{p}/2$ different ways to choose two vectors $\mathbf{x}, \mathbf{x}'$ of weight p in the first k positions. This means that the probability that there exists at least one pair of vectors $\mathbf{x}, \mathbf{x}'$ with $\phi^{\mathbf{Z}}(\mathbf{x}) = \phi^{\mathbf{Z}}(\mathbf{x}') = \mathbf{0}$, denoted here by $\mathcal{Q}$, is
$$\mathcal{Q} = 1 - \left(1 - 2^{-z}\right)^{\binom{2p}{p}/2}. \tag{6}$$

We may now put all the expressions together to get the overall avarage complexity for finding the low weight codeword.

Theorem 1 *The total complexity of the algorithm is given by*

$$\frac{\binom{n}{w}\left((n-k)^2(n+k)/2 + \binom{k}{p/2}pz/2 + \binom{k}{p}2^{-z}pl + \left(\binom{k}{p}2^{-z}\right)^2 2^{-l}2p(n-k)\right)}{\binom{k}{2p}\binom{n-k-z-l}{w-2p}\left(1-(1-2^{-z})^{\binom{2p}{p}/2}\right)}.$$

Proof. Putting together the parts we obtained, we get
$$\mathbf{E}\left[\#\text{operations}\right] = \frac{A_0 + A_1 + A_2 + A_3}{\mathcal{W}\mathcal{Q}}$$
$$= \frac{\binom{n}{w}\left((n-k)^2(n+k)/2 + \binom{k}{p/2}pz/2 + \binom{k}{p}2^{-z}pl + \left(\binom{k}{p}2^{-z}\right)^2 2^{-l}2p(n-k)\right)}{\binom{k}{2p}\binom{n-k-z-l}{w-2p}\left(1-(1-2^{-z})^{\binom{2p}{p}/2}\right)}.$$

4.1 Parameters and security

Various examples of parameters were given in [1] to obtain certain security levels. New bit security levels of our new algorithm applied the given parameters are presented in Table 1.

As for the case $(6624, 5129, 117)$, that is recommended for 256 bit security, we break it with on average $2^{247.29}$ operations with $p = 12$, $z = 52$ and $l = 74$. Moreover, for the 1000 bit security level with parameters $(30332, 22968, 494)$, we obtain a much better complexity of $2^{976.55}$ operations using our algorithm with $p = 28$, $z = 53$ and $l = 264$.

Parameters			Bit security		
n	k	w	Original Stern	Improved Stern	Ball-collision
1632	1269	34	82.23	78.91	81.33
2960	2288	57	129.84	125.06	127.89
6624	5129	117	258.61	247.29	254.15
30332	22968	494	1007.4	976.55	996.22

Table 1. Choice of parameters and their security with corresponding algorithm.

4.2 Further improvements

Adjusting coefficients There are several improvements that can be applied to the coefficients c_1, c_2 and c_3. For instance, if we in the first step sum the both the z bits of the first field and the l bits from the second field, then in the second step we only need to sum two l-bit vectors. In order to balance out the complexity of the steps, one can calculate a subset of the positions in l, as given below:

$$\mathcal{K}(m) \stackrel{\text{def}}{=} (c_1, c_2, c_3) = (p(z+m)/2, p(l-m) + 2m, 2p(n-k))$$

To find the best complexity, we minimize over both p and m.

5 Conclusions

In this article we have proposed a new information set decoding algorithm based on Stern's algorithm which further pushes down the complexity of decoding random codes. We have specified parameters for expected numbers of bit operations. Also, we have proposed minor improvements that can be applied to ISD algorithms in general.

References

1. D. J. Bernstein, T. Lange, and C. Peters. Smaller decoding exponents: Ball collision decoding. volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer-Verlag, 2011.
2. A. Canteaut and 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 Transactions on Information Theory*, 44:367–378, 1998.
3. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer-Verlag, 2009.
4. N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, editor, *Advances in Cryptology—EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer-Verlag, 2010.
5. J. Stern. A method for finding codewords of small weight. In J. Wolfmann G. D. Cohen, editor, *Coding theory and applications*, Lecture Notes in Computer Science, pages 106–113. Springer-Verlag, 1989.
6. P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology—EUROCRYPT'89*.
7. J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. volume 34 of *IEEE Transactions on Information Theory*, pages 1354–1359, 1988.
8. A. May, A. Meurer, and E. Tomae. Decoding random linear codes in $\mathcal{O}(2^{0.054n})$. *http://www.cits.rub.de/imperia/md/content/may/paper/decoding.pdf*.
9. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report 42–44*, pages 114–116, 1978.
10. E. Prange. The use of information sets in decoding cyclic codes. In *IRE Transactions on Information Theory IT-8*, pages 5–9, 1962.
11. V. Sidelnikov and S. Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. In *Discrete Math. Appl.*, pages 439–444. Springer-Verlag, 1992.
12. J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology—CRYPTO'93*, volume 388 of *Lecture Notes in Computer Science*, pages 13–21. Springer-Verlag, 1994.