Smaller decoding exponents: ball-collision decoding

Daniel J. Bernstein[1], Tanja Lange[2], and Christiane Peters[2]

[1] Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
tanja@hyperelliptic.org, c.p.peters@tue.nl

Abstract. Very few public-key cryptosystems are known that can encrypt and decrypt in time $b^{2+o(1)}$ with conjectured security level 2^b against conventional computers and quantum computers. The oldest of these systems is the classic McEliece code-based cryptosystem.

The best attacks known against this system are generic decoding attacks that treat McEliece's hidden binary Goppa codes as random linear codes. A standard conjecture is that the best possible w-error-decoding attacks against random linear codes of dimension k and length n take time $2^{(\alpha(R,W)+o(1))n}$ if $k/n \to R$ and $w/n \to W$ as $n \to \infty$.

Before this paper, the best upper bound known on the exponent $\alpha(R, W)$ was the exponent of an attack introduced by Stern in 1989. This paper introduces "ball-collision decoding" and shows that it has a smaller exponent for each (R, W): the speedup from Stern's algorithm to ball-collision decoding is exponential in n.

Keywords: McEliece cryptosystem, Niederreiter cryptosystem, post-quantum cryptography, attacks, information-set decoding, collision decoding

1 Introduction

In 1978, McEliece introduced a code-based public-key cryptosystem that has maintained remarkable strength against every proposed attack. The top threats against McEliece's system have always been generic decoding algorithms that decode random linear codes. The standard conjecture is that the best possible generic decoding algorithm takes exponential time for any constant asymptotic code rate R and constant asymptotic error fraction W: i.e., time $2^{(\alpha(R,W)+o(1))n}$ for some positive real number $\alpha(R, W)$ if $k/n \to R$ and $w/n \to W$ as $n \to \infty$. Here n is the code length, k is the code dimension, and w is the number of errors.

Permanent ID of this document: 0e8c929565e20cf63e6a19794e570bb1. Date: 2011.05.27. This work was supported by the Cisco University Research Program, by the National Institute of Standards and Technology under grant 60NANB10D263, and by the European Commission under Contract ICT-2007-216646 ECRYPT II.

Two decades ago a flurry of fundamental algorithmic improvements produced a new upper bound on the optimal decoding exponent $\alpha(R, W)$. The upper bound is the exponent of a 1989 algorithm by Stern [50]. This upper bound arises from an asymptotic binomial-coefficient optimization and does not have a simple formula, but it can be straightforwardly computed to high precision for any particular (R, W). For example, for $W = 0.04$ and $R = 1 + W \log_2 W + (1 - W) \log_2(1-W) = 0.7577\ldots$, Stern's algorithm shows that $\alpha(R, W) \leq 0.0809\ldots$.

There have also been many polynomial-factor speedups in generic decoding algorithms; there are dozens of papers on this topic, both inside and outside cryptography. Here is an illustration of the cumulative impact of many of the speedups. McEliece's original parameter suggestions ("$n = 1024, k = 524, t = 50$") take about $524^3\binom{1024}{50}/\binom{500}{50} \approx 2^{81}$ operations to break by the simple information-set-decoding attack explained in McEliece's original paper [41, Section 3]. (McEliece estimated the attack cost as $524^3(1 - 50/1024)^{-524} \approx 2^{65}$; this underestimate was corrected by Adams and Meijer in [2, Section 3].) The attack we presented in [8], thirty years after McEliece's paper, builds on several improvements and takes only about $2^{60.5}$ operations for the same parameters. That attack was carried out successfully, decrypting a challenge ciphertext. More recent improvements include [28] and [45]; see Section 4 for a more comprehensive discussion of the literature.

However, polynomial factors are asymptotically $2^{o(n)}$, and thus have no relevance to the exponent $\alpha(R, W)$ in $2^{(\alpha(R,W)+o(1))n}$. The best known upper bound on $\alpha(R, W)$ has been unchanged since 1989.

Contents of this paper. This paper presents smaller upper bounds on the decoding exponent $\alpha(R, W)$. Specifically, this paper introduces a generic decoding algorithm and demonstrates that this algorithm is, for every (R, W), faster than Stern's algorithm by a factor exponential in n. We call this algorithm "ball-collision decoding" because of a geometric interpretation explained in Section 4. The change in the exponent is not very large — for example, this paper uses ball-collision decoding to demonstrate that $\alpha(R, W) \leq 0.0807\ldots$ for the (R, W) given above — but it is the first exponential improvement in decoding complexity in more than twenty years.

This paper also evaluates the exact cost of ball-collision decoding, using the same bit-operation-counting rules as in the previous literature, and uses this evaluation to illustrate the impact of ball-collision decoding upon cryptographic applications. For example, the parameters $(6624, 5129, 117)$ were proposed in [8, Section 7] at a 256-bit security level against a state-of-the-art refinement of Stern's algorithm; this paper shows that ball-collision decoding costs 2.6 times fewer bit operations. At a theoretical 1000-bit security level the improvement grows to 15.5. These concrete figures are consistent with the asymptotic analysis.

Of course, actually breaking these parameters remains very far out of reach, and these results should not be interpreted as damaging the viability of the McEliece cryptosystem. However, these results *do* raise new questions regarding the proper choice of parameters for the McEliece cryptosystem. Section 8 discusses the problem of parameter selection for code-based cryptography.

We also wrote a straightforward reference implementation of ball-collision decoding, and tested the implementation on a long series of random challenges at a much lower security level. The costs and success probabilities observed in these experiments matched the formulas shown in this paper.

Attack model. "Attacks" above refer only to passive single-target inversion attacks. The original McEliece cryptosystem, like the original RSA cryptosystem, is really just a trapdoor one-way function; when used naively as a public-key cryptosystem it is trivially broken by chosen-ciphertext attacks such as Berson's attack [11] and the Verheul–Doumen–van Tilborg attack [53].

Protecting the McEliece system against these attacks, to meet the standard notion of IND-CCA2 security for a public-key cryptosystem, requires appropriate padding and randomization, similar to RSA-OAEP. As shown by Kobara and Imai in [36], adding this protection does not significantly increase the cost of the McEliece cryptosystem.

2 Review of the McEliece cryptosystem

The public key in the McEliece cryptosystem consists of a random-looking rank-k matrix $G \in \mathbf{F}_2^{k \times n}$. The sender encrypts a message m in $\mathbf{F}_2^k$ by first multiplying it with the matrix G, producing mG; choosing uniformly at random a word e in $\mathbf{F}_2^n$ of Hamming weight w; and adding e to mG, producing a ciphertext $mG + e$. The cryptosystem parameters are n, k, w.

The legitimate receiver decrypts $mG + e$ using a secret key which consists of a secret decoding algorithm producing the *error vector* e given $mG + e$. The details are not relevant to our attack and can be found in, e.g., [44].

An attacker is faced with the problem of determining e given G and $mG + e$. Note that finding e is equivalent to finding the message m: subtracting e from $mG + e$ produces mG, and then simple linear transformations produce m.

The set $\mathbf{F}_2^k G = \left\{ mG : m \in \mathbf{F}_2^k \right\}$ is called a *linear code* of *length* n and *dimension* k, specifically the linear code *generated by* G. The matrix G is called a *generator matrix* for this code. The elements of $\mathbf{F}_2^k G$ are called *codewords*. If the linear code $\mathbf{F}_2^k G$ equals $\{c \in \mathbf{F}_2^n : Hc = 0\}$ then the matrix H is called a *parity-check matrix* for the code.

Without loss of generality one can assume that the matrix G in a CCA2-secure version of the McEliece cryptosystem is given in *systematic form* $G = (I_k | -A^T)$ where I_k is a $k \times k$ identity matrix and A an $(n-k) \times k$ matrix. Then the matrix $H = (A | I_{n-k})$ is a parity-check matrix for the code generated by G.

An *information set* Z for H is a set of k integers in $\{1, 2, \ldots, n\}$ for which the $n - k$ columns of H that are not indexed by Z are linearly independent. Applying Gaussian elimination to those $n - k$ columns shows that codewords are determined by their Z-indexed components. For example, $\{1, 2, \ldots, k\}$ is an information set for $H = (A | I_{n-k})$.

Fix $m \in \mathbf{F}_2^k$ and $e \in \mathbf{F}_2^n$ with $\mathrm{wt}(e) = w$. Write $c = mG$. By linearity one has $H(c + e) = Hc + He = He$ since $Hc = 0$. The result $s = He$ is called the

syndrome of e. It is the sum of the w columns of H indexed by the positions of 1's in e. The attacker's task is equivalent to finding e given H and $s = He$.

3 The ball-collision-decoding algorithm

This section introduces ball-collision decoding. It first states the algorithm and then discusses various optimizations. Section 4 explains how this algorithm relates to previous algorithms.

The algorithm is given a parity-check matrix $H \in \mathbf{F}_2^{(n-k)\times n}$, a syndrome $s \in \mathbf{F}_2^{n-k}$, and a weight $w \in \{0, 1, 2, \ldots\}$. The goal of the algorithm is to find a corresponding error vector e: i.e., a vector $e \in \mathbf{F}_2^n$ of weight w such that $s = He$.

Ball-collision decoding has its roots in information-set decoding, which was used against the McEliece system in, e.g., [50], [17], [18], and [8]. The previous algorithms select a random information set in the parity-check matrix and then search for vectors having a particular pattern of non-zero entries. Ball-collision decoding is similar but searches for a more complicated, and more likely, pattern. See Section 4 for further discussion of the previous work.

The reader is encouraged to consider, while reading the algorithm, the case that the algorithm is given a matrix H already in systematic form and that it chooses $Z = \{1, 2, \ldots, k\}$ as information set. The matrix U in Step 4 is then the identity matrix I_{n-k}. The algorithm divides H into blocks, and divides the syndrome s into corresponding blocks, as specified by algorithm parameters ℓ_1, ℓ_2:

$$H = \begin{pmatrix} A_1 & I_1 & 0 \\ A_2 & 0 & I_2 \end{pmatrix}, \qquad s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix},$$

where $s_1 \in \mathbf{F}_2^{\ell_1+\ell_2}$, $s_2 \in \mathbf{F}_2^{n-k-\ell_1-\ell_2}$, $A_1 \in \mathbf{F}_2^{(\ell_1+\ell_2)\times k}$, $A_2 \in \mathbf{F}_2^{(n-k-\ell_1-\ell_2)\times k}$, and each I_i is an identity matrix.

One iteration of ball-collision decoding:
CONSTANTS: $n, k, w \in \mathbf{Z}$ with $0 \le w \le n$ and $0 \le k \le n$.
PARAMETERS: $p_1, p_2, q_1, q_2, k_1, k_2, \ell_1, \ell_2 \in \mathbf{Z}$ with $0 \le k_1$, $0 \le k_2$, $k = k_1 + k_2$,
$0 \le p_1 \le k_1$, $0 \le p_2 \le k_2$, $0 \le q_1 \le \ell_1$, $0 \le q_2 \le \ell_2$,
and $0 \le w - p_1 - p_2 - q_1 - q_2 \le n - k - \ell_1 - \ell_2$.
INPUT: $H \in \mathbf{F}_2^{(n-k)\times n}$ and $s \in \mathbf{F}_2^{n-k}$.
OUTPUT: Zero or more vectors $e \in \mathbf{F}_2^n$ with $He = s$ and $\mathrm{wt}(e) = w$.

1. Choose a uniform random information set Z. Subsequent steps of the algorithm write "$\mathbf{F}_2^Z$" to refer to the subspace of $\mathbf{F}_2^n$ supported on Z.
2. Choose a uniform random partition of Z into parts of sizes k_1 and k_2. Subsequent steps of the algorithm write "$\mathbf{F}_2^{k_1}$" and "$\mathbf{F}_2^{k_2}$" to refer to the corresponding subspaces of $\mathbf{F}_2^Z$.
3. Choose a uniform random partition of $\{1, 2, \ldots, n\} \setminus Z$ into parts of sizes ℓ_1, ℓ_2, and $n - k - \ell_1 - \ell_2$. Subsequent steps of the algorithm write "$\mathbf{F}_2^{\ell_1}$" and "$\mathbf{F}_2^{\ell_2}$" and "$\mathbf{F}_2^{n-k-\ell_1-\ell_2}$" to refer to the corresponding subspaces of $\mathbf{F}_2^{\{1,2,\ldots,n\}\setminus Z}$.

4. Find an invertible $U \in \mathbf{F}_2^{(n-k)\times(n-k)}$ such that the columns of UH indexed by $\{1,2,\ldots,n\}\setminus Z$ are an $(n-k)\times(n-k)$ identity matrix. Write the columns of UH indexed by Z as $\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ with $A_1 \in \mathbf{F}_2^{(\ell_1+\ell_2)\times k}$, $A_2 \in \mathbf{F}_2^{(n-k-\ell_1-\ell_2)\times k}$.
5. Write Us as $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$ with $s_1 \in \mathbf{F}_2^{\ell_1+\ell_2}$, $s_2 \in \mathbf{F}_2^{n-k-\ell_1-\ell_2}$.
6. Compute the set S consisting of all triples $(A_1x_0+x_1, x_0, x_1)$ where $x_0 \in \mathbf{F}_2^{k_1}$, $\mathrm{wt}(x_0) = p_1$, $x_1 \in \mathbf{F}_2^{\ell_1}$, $\mathrm{wt}(x_1) = q_1$.
7. Compute the set T consisting of all triples $(A_1y_0 + y_1 + s_1, y_0, y_1)$ where $y_0 \in \mathbf{F}_2^{k_2}$, $\mathrm{wt}(y_0) = p_2$, $y_1 \in \mathbf{F}_2^{\ell_2}$, $\mathrm{wt}(y_1) = q_2$.
8. For each $(v, x_0, x_1) \in S$:
   For each y_0, y_1 such that $(v, y_0, y_1) \in T$:
   If $\mathrm{wt}(A_2(x_0 + y_0) + s_2) = w - p_1 - p_2 - q_1 - q_2$:
   Output $x_0 + y_0 + x_1 + y_1 + A_2(x_0 + y_0) + s_2$.

Note that Step 8 is a standard "join" operation between S and T; it can be implemented efficiently by sorting, by hashing, or by simple table indexing. In [8, Section 6] we describe an efficient implementation of essentially the same operation using only about $2^{\ell_1+\ell_2+1}$ bits of memory. See Sections 5 and 6 for further discussion of arithmetic costs and memory-access costs.

Theorem 3.1 (Correctness of ball-collision decoding) *The set of output vectors e of the ball-collision decoding algorithm is the set of vectors e that satisfy $He = s$ and have weights $p_1, p_2, q_1, q_2, w - p_1 - p_2 - q_1 - q_2$ in $\mathbf{F}_2^{k_1}$, $\mathbf{F}_2^{k_2}$, $\mathbf{F}_2^{\ell_1}$, $\mathbf{F}_2^{\ell_2}$, and $\mathbf{F}_2^{n-k-\ell_1-\ell_2}$ respectively.*

Proof. Each element $(v, x_0, x_1) \in S$ satisfies $x_0 \in \mathbf{F}_2^{k_1}$ with $\mathrm{wt}(x_0) = p_1$; $v = A_1x_0 + x_1$ and $x_1 \in \mathbf{F}_2^{\ell_1}$ with $\mathrm{wt}(x_1) = q_1$. Similarly each element $(v, y_0, y_1) \in T$ satisfies $y_0 \in \mathbf{F}_2^{k_2}$ with $\mathrm{wt}(y_0) = p_2$; $v = A_1y_0 + y_1 + s_1$; $y_1 \in \mathbf{F}_2^{\ell_2}$ with $\mathrm{wt}(y_1) = q_2$. Now, with Z-indexed columns visualized as coming before the remaining columns, we have

$$UHe = UH \begin{pmatrix} x_0+y_0 \\ x_1+y_1 \\ A_2(x_0+y_0)+s_2 \end{pmatrix} = \begin{pmatrix} A_1(x_0+y_0)+x_1+y_1 \\ A_2(x_0+y_0)+A_2(x_0+y_0)+s_2 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = Us$$

so $He = s$. Furthermore, $x_0 + y_0 \in \mathbf{F}_2^{k_1+k_2}$ has weights p_1, p_2 in $\mathbf{F}_2^{k_1}, \mathbf{F}_2^{k_2}$; $x_1 + y_1 \in \mathbf{F}_2^{\ell_1+\ell_2}$ has weights q_1, q_2 in $\mathbf{F}_2^{\ell_1}, \mathbf{F}_2^{\ell_2}$; and $\mathrm{wt}(A_2(x_0+y_0)+s_2) = w-p_1-p_2-q_1-q_2$.

Conversely, the iteration finds every vector e having this weight distribution and satisfying $He = s$. Indeed, write e as $x_0 + y_0 + x_1 + y_1 + e_2$ with $x_0 \in \mathbf{F}_2^{k_1}$, $y_0 \in \mathbf{F}_2^{k_2}$, $x_1 \in \mathbf{F}_2^{\ell_1}$, $y_1 \in \mathbf{F}_2^{\ell_2}$, and $e_2 \in \mathbf{F}_2^{n-k-\ell_1-\ell_2}$. By hypothesis the weights of x_0, y_0, x_1, y_1, e_2 are $p_1, p_2, q_1, q_2, w - p_1 - p_2 - q_1 - q_2$, respectively. Now define $v = A_1x_0 + x_1$. The equation $UHe = Us$ implies $v = A_1y_0 + y_1 + s_1$; and $e_2 = A_2(x_0 + y_0) + s_2$. Hence $(v, x_0, x_1) \in S$ and $(v, y_0, y_1) \in T$. Finally $\mathrm{wt}(A_2(x_0 + y_0) + s_2) = \mathrm{wt}(e_2) = w - p_1 - p_2 - q_1 - q_2$ so the algorithm prints e as claimed. □

Finding an information set. The simplest way to choose a uniform random information set is to repeatedly choose a uniform random size-k subset $Z \subseteq \{1, 2, \ldots, n\}$ until the $n - k$ columns of H indexed by $\{1, 2, \ldots, n\} \setminus Z$ are linearly independent. Standard practice (see, e.g., Stern [50]) is to eliminate the fruitless Gaussian-elimination steps here, at the expense of negligible bias, by assembling the information set one column at a time, ensuring that each newly added column is linearly independent of the previously selected columns. After this optimization there is only one Gaussian-elimination step per iteration.

Reusing intermediate sums. Computing the vector A_1x_0 for a weight-p_1 word x_0 in $\mathbf{F}_2^{k_1}$ can be done by adding the specified p_1 columns of A_1 in $p_1 - 1$ additions in $\mathbf{F}_2^{\ell_1+\ell_2}$.

Computing A_1x_0 for *all* the $\binom{k_1}{p_1}$ vectors x_0 can be done more efficiently than repeating this process for each of them. Start by computing all $\binom{k_1}{2}$ sums of 2 columns of A_1; each sum costs one addition in $\mathbf{F}_2^{\ell_1+\ell_2}$. Then compute all $\binom{k_1}{3}$ sums of 3 columns of A_1 by adding one extra column to the previous results. Proceed in the same way until all $\binom{k_1}{p_1}$ sums of p_1 columns of A_1 are computed. This produces all required sums in only marginally more than one $\mathbf{F}_2^{\ell_1+\ell_2}$ addition per sum; see Section 5 for a precise operation count.

Early abort. The vector $A_2(x_0 + y_0) + s_2$ is computed as a sum of $p_1 + p_2 + 1$ vectors of length $n-k-\ell_1-\ell_2$. Instead of computing the sum on all $n-k-\ell_1-\ell_2$ positions one computes the sum row by row and simultaneously checks the weight. If the weight exceeds $w-p_1-p_2-q_1-q_2$ one can discard this particular pair (x_0, y_0).

We comment that one can further reduce the cost of this step by precomputing sums of smaller sets of columns, but we do not use this idea in our analysis, because it is not critical for the algorithm's performance.

4 Relationship to previous algorithms

This section discusses the relationship of ball-collision decoding to previous information-set-decoding algorithms.

Collision decoding vs. ball-collision decoding. We use the name "collision decoding" for the special case $q_1 = q_2 = 0$ of ball-collision decoding. The idea of collision decoding is more than twenty years old: Stern's algorithm in [50] is, aside from trivial details, exactly the special case $q_1 = q_2 = 0$, $p_1 = p_2$, $k_1 \approx k_2$. Dumer in [26] independently introduced the core idea, although in a more limited form, and in [27] achieved an algorithm similar to Stern's.

All state-of-the-art decoding attacks since [50] have been increasingly optimized forms of collision decoding. Other approaches to decoding, such as "gradient decoding" ([4]), "supercode decoding" ([5]), and "statistical decoding" (see [3] and [43]), have never been competitive with Stern's algorithm. This does not mean that those approaches should be ignored; our generalization from collision

decoding to ball-collision decoding is inspired by one of the steps in supercode decoding.

Collision decoding searches for collisions in $\mathbf{F}_2^{\ell_1+\ell_2}$ between points A_1x_0 and points $A_1y_0 + s_1$. Ball-collision decoding expands each point A_1x_0 into a small ball (in the Hamming metric), namely $\{A_1x_0 + x_1 : x_1 \in \mathbf{F}_2^{\ell_1}, \mathrm{wt}(x_1) = q_1\}$; similarly expands each point A_1y_0 into a small ball; and searches for collisions between these balls.

From the perspective of ball-collision decoding, the fundamental disadvantage of collision decoding is that errors are required to avoid an asymptotically quite large stretch of $\ell_1 + \ell_2$ positions. Ball-collision decoding makes a much more reasonable hypothesis, namely that there are asymptotically increasingly many errors in those positions. It requires extra work to enumerate the points in each ball, but the extra work is only about the square root of the improvement in success probability. The cost ratio is exponential when all parameters are optimized properly; see Section 7.

Collision decoding also has a secondary disadvantage compared to ball-collision decoding: its inner loop is slower, since computing A_1x_0 for a new x_0 is considerably more expensive than adding x_1 for a new x_1. The cost ratio here is only polynomial, and is not relevant to the exponents (see Section 7), but is accounted for in the bit-operation count (see Section 5). This disadvantage of collision decoding is also visible in the number of memory accesses to A_1 (see Section 6); however, standard practice in the literature on this topic is to count the number of bit operations involved in arithmetic and to ignore the cost of memory access.

Additional credits. The simplest form of information-set decoding, introduced by Prange in [47], did not allow errors in the information set. For asymptotic analyses see [41], [1], and [2].

The idea of allowing errors was published by Lee and Brickell in [38], by Leon in [39], and by Krouk in [37], but without Stern's collision idea; in the terminology of ball-collision decoding, with $p_2 = 0$, $q_1 = q_2 = 0$, and $\ell_2 = 0$. For each pattern of p_1 errors in k columns, Lee and Brickell checked the weight of the remaining $n - k$ columns; Leon and Krouk required ℓ_1 columns to have weight 0, and usually checked only those columns. For asymptotic analyses see [37], [23], and [24].

Overbeck and Sendrier [44] give a visual comparison of the algorithms by comparing to which interval they restrict how many errors. Figure 4.1 extends their picture to include ball-collision decoding. It shows that the new algorithm allows errors in an interval that had to be error-free in Leon's and Stern's algorithms.

The idea of allowing errors everywhere can be extracted, with considerable effort, from the description of supercode decoding in [5]. After a detailed analysis we have concluded that the algorithm in [5] is much slower than collision decoding. The same algorithm is claimed in [5] to have smaller exponents than collision decoding (with astonishing gaps, often 15% or more), but this claim is based on a chain of exponentially large inaccuracies in the algorithm analysis in [5]. The starting point of the chain is [5, "Corollary 12"], which claims size $\binom{k}{e_1}\binom{y}{e_2}/2^{by}$ for lists that actually have size $\binom{k}{e_1}\binom{y}{e_2}^b/2^{by}$.

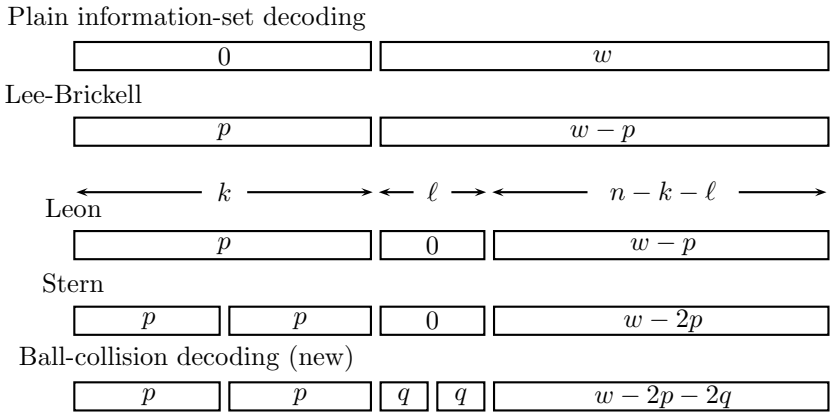

Fig. 4.1. Error positions hypothesized by various decoding algorithms.

The idea of allowing errors everywhere can also be found in the much more recent paper [28], along with a polynomial-factor "birthday" speedup obtained by dropping Stern's left-right separation. The algorithm analysis by Finiasz and Sendrier in [28] concludes that the overall "gain compared with Stern's algorithm" is a constant times "$\sqrt[4]{\pi p/2}$", which is bounded by a polynomial in n. Our own assessment is that if parameters had been chosen more carefully then the algorithm of [28] would have led to an exponential improvement over collision decoding, contrary to the conclusions in [28]. This algorithm would still have retained the secondary disadvantage described above, and therefore would not have been competitive with ball-collision decoding.

A more detailed analysis of the "birthday" speedup in collision decoding appeared in [45] along with an optimized generalization to $\mathbf{F}_q$. These modifications can be adapted to ball-collision decoding but would complicate the algorithm statement and analysis without changing the exponent of binary decoding; we have skipped these modifications for simplicity.

One way to speed up Gaussian elimination is to change only one information-set element in each iteration. This idea was introduced by Omura, according to [22, Section 3.2.4]. It was applied to increasingly optimized forms of information-set decoding by van Tilburg in [51] and [52], by Chabanne and Courteau in [19], by Chabaud in [20], by Canteaut and Chabanne in [16], by Canteaut and Chabaud in [17], and by Canteaut and Sendrier in [18]. In [8] we improved the balance between Gaussian-elimination cost and error-searching cost by changing c information-set elements in each iteration for an optimized value of c. The ideas of reusing sums and aborting weight calculations also appeared in [8], in the context of an improved collision-decoding algorithm.

5 Complexity analysis

This section analyzes the complexity of ball-collision decoding. In particular, this section analyzes the success probability of each iteration and the number of bit operations needed for each iteration.

Success probability. Assume that e is a uniform random vector of weight w. One iteration of ball-collision decoding finds e exactly if it has the right weight distribution, namely weight p_1 in the first k_1 positions specified by the information set, weight p_2 in the remaining k_2 positions specified by the information set, weight q_1 on the first ℓ_1 positions outside the information set, and weight q_2 on the next ℓ_2 positions outside the information set.

The probability that e has this weight distribution is, by a simple counting argument, exactly

$$b(p_1,p_2,q_1,q_2,\ell_1,\ell_2) = \binom{n}{w}^{-1}\binom{n-k-\ell_1-\ell_2}{w-p_1-p_2-q_1-q_2}\binom{k_1}{p_1}\binom{k_2}{p_2}\binom{\ell_1}{q_1}\binom{\ell_2}{q_2}.$$

The expected number of iterations of the outer loop is, for almost all H, very close to the reciprocal of the success probability of a single iteration. We explicitly disregard, without further comment, the extremely unusual codes for which the average number of iterations is significantly different from the reciprocal of the success probability of a single iteration. For further discussion of this issue and how unusual it is see, e.g., [24] and [10].

Gaussian elimination. There are many ways to speed up Gaussian elimination, as discussed in Section 4; implementors are encouraged to use those optimizations. However, in this paper we will be satisfied with a quite naive form of Gaussian elimination, taking $(1/2)(n-k)^2(n+k)$ bit operations; our interest is in large input sizes, and elimination takes negligible time for those sizes.

Building the set S. The total cost of computing A_1x_0 for all x_0 of Hamming weight p_1, using intermediate sums as explained in Section 3, is

$$(\ell_1+\ell_2)\left(\binom{k_1}{2}+\binom{k_1}{3}+\cdots\binom{k_1}{p_1}\right).$$

Using $L(k,p) = \sum_{i=1}^{p}\binom{k}{i}$ as a shorthand, the costs can be written as $(\ell_1+\ell_2)\,(L(k_1,p_1)-k_1)$. The $\ell_1+\ell_2$ factor is the number of bit operations to compute A_1x_0 from A_1x_0' where x_0 extends x_0' by a single bit.

Then for each x_0 all $\binom{\ell_1}{q_1}$ possible words x_1 in $\mathbf{F}_2^{\ell_1}$ of weight q_1 are added to A_1x_0, producing $A_1x_0+x_1$. For x_1, as for x_0, we loop over the possible sets of indices, and reuse sums obtained from subsets. This slightly increases the number of sums up to $L(\ell_1,q_1)$, but decreases the cost of each sum down to a single bit operation, computing $A_1x_0+x_1$ from $A_1x_0+x_1'$. Overall this step takes $\min\{1,q_1\}\binom{k_1}{p_1}L(\ell_1,q_1)$ bit operations; note that for $q_1=0$ the cost of this step is indeed 0.

Each choice of (x_0,x_1) adds one element to S. Hence, the number of elements in S equals exactly the number of choices for x_0 and x_1, i.e. $\#S = \binom{k_1}{p_1}\binom{\ell_1}{q_1}$.

Building the set T. The set T is built similarly to the set S. The only difference is that the expression $A_1y_0 + y_1 + s_1$ involves adding s_1 and thus the single columns (corresponding to weight-1 words y_0) already cost $(\ell_1 + \ell_2)\binom{k_2}{1}$ bit operations. In total this step takes $(\ell_1 + \ell_2)L(k_2, p_2) + \min\{1, q_2\}\binom{k_2}{p_2}L(\ell_2, q_2)$.

The set T contains exactly $\#T = \binom{k_2}{p_2}\binom{\ell_2}{q_2}$ elements.

Checking collisions. The last step does one check for every (x_0, x_1, y_0, y_1) satisfying the equation $A_1x_0 + x_1 = A_1y_0 + y_1 + s_1$. There are $\binom{k_1}{p_1}\binom{k_2}{p_2}\binom{\ell_1}{q_1}\binom{\ell_2}{q_2}$ choices of (x_0, x_1, y_0, y_1).

If the vectors v appearing in S and T were uniformly distributed among the $2^{\ell_1+\ell_2}$ possible values then on average $\#S \cdot \#T \cdot 2^{-\ell_1-\ell_2}$ checks would be done. The expected number of checks is extremely close to this for almost all H; as above we disregard the extremely unusual codes with different behavior.

Each check consists of computing $\mathrm{wt}(A_2(x_0 + y_0) + s_2)$ and testing whether it equals $w - p_1 - p_2 - q_1 - q_2$. When using the early-abort weight calculation, on average only $2(w - p_1 - p_2 - q_1 - q_2 + 1)$ bits of the result are computed before the weight is found too high. Each bit of the result costs $p_1 + p_2$ bit operations because $x_0 + y_0$ has weight $p_1 + p_2$.

Cost of one iteration. To summarize, the total cost per iteration of the inner loop with parameters $p_1, p_2, q_1, q_2, \ell_1, \ell_2$ amounts to

$$\begin{aligned} &c(p_1, p_2, q_1, q_2, \ell_1, \ell_2) \\ &\quad = \frac{1}{2}(n-k)^2(n+k) + (\ell_1 + \ell_2)\big(L(k_1, p_1) + L(k_2, p_2) - k_1\big) \\ &\qquad + \min\{1, q_1\}\binom{k_1}{p_1}L(\ell_1, q_1) + \min\{1, q_2\}\binom{k_2}{p_2}L(\ell_2, q_2) \\ &\qquad + 2(w - p_1 - p_2 - q_1 - q_2 + 1)(p_1 + p_2)\binom{k_1}{p_1}\binom{k_2}{p_2}\binom{\ell_1}{q_1}\binom{\ell_2}{q_2}2^{-\ell_1-\ell_2}. \end{aligned}$$

6 Concrete parameter examples

This section considers concrete examples in order to show the speedup gained by ball-collision decoding in comparison to collision decoding. The first parameters were previously proposed to achieve 256-bit security against current attacks. We designed the second parameters according to similar rules to achieve a 1000-bit security level against current attacks. We do not mean to suggest that 1000-bit security is of any real-world relevance; we consider it to illustrate the asymptotic superiority of ball-collision decoding.

Finiasz and Sendrier in [28] presented "lower bounds on the effective work factor of existing real algorithms, but also on the future improvements that could be implemented"; and said that beating these bounds would require the introduction of "new techniques, never applied to code-based cryptosystems". For each set of parameters we evaluate the Finiasz–Sendrier lower bound and the costs of three algorithms:

(1) collision decoding ($q_1 = q_2 = 0$),
(2) collision decoding using the birthday trick from [28] as analyzed in [45], and
(3) ball-collision decoding.

Ball-collision decoding beats the Finiasz–Sendrier lower bound in both of these examples. The main reason for this is that ball-collision decoding dodges the secondary disadvantage described in Section 4; the lower bound assumes that each new vector requires $\ell_1 + \ell_2$ bit operations to update A_1x_0, but in ball-collision decoding each new vector requires just 1 bit operation to update x_1.

We emphasize that all of these costs and bounds use the same model of computation, counting the number of bit operations for arithmetic and disregarding costs of memory access, copies, etc. A table-indexing join operation can easily be carried out for free in this model. We would prefer a more carefully defined model of computation that includes realistic memory-access costs, such as the Brent–Kung circuit model [13], but the bit-operation model is simpler and is standard in papers on this topic.

256-security revisited. According to [8, Section 7] a binary code with length $n = 6624$, $k = 5129$, $w = 117$ achieves 256-bit security. The best collision-decoding parameters are actually slightly below 2^{256} bit operations: they use $2^{181.4928}$ iterations (on average), each taking $2^{74.3741}$ bit operations, for a total of $2^{255.8669}$ bit operations.

Collision decoding with the birthday trick takes, with optimal parameters, $2^{255.54880}$ bit operations. The birthday trick increases the cost per iteration by a factor of 2.2420 compared to the classical collision-decoding algorithm, to $2^{75.5390}$ bit operations. However, the trick increases the chances of finding the desired error vector noticeably, reducing the number of iterations by a factor of 2.7951, to $2^{180.0099}$. Thus the birthday trick yields an overall 1.2467× speedup.

The Finiasz–Sendrier lower bound is $2^{255.1787}$ bit operations, 1.6112× smaller than the cost of collision decoding.

Ball-collision decoding with parameters $k_1 = 2565$, $k_2 = 2564$, $\ell_1 = \ell_2 = 47$, $p_1 = p_2 = 8$, and $q_1 = q_2 = 1$ needs only $2^{254.1519}$ bit operations to attack the same system. On average the algorithm needs $2^{170.6473}$ iterations each taking $2^{83.5046}$ bit operations.

Ball-collision decoding thus costs 3.2830× less than collision decoding, 2.6334× less than collision decoding with the birthday trick, and 2.0375× less than the Finiasz–Sendrier lower bound.

1000-bit security. Attacking a system based on a code of length $n = 30332$, $k = 22968$, $w = 494$ requires $2^{1000.9577}$ bit operations using collision decoding with the optimal parameters $k_1 = k_2 = 11484$, $\ell_1 = \ell_2 = 140$, $p_1 = p_2 = 27$ and $q_1 = q_2 = 0$.

The birthday trick reduces the cost by a factor of 1.7243, to $2^{1000.1717}$ bit operations. This means that this system offers 1000-bit security against all previously known attacks.

The Finiasz–Sendrier lower bound is $2^{999.45027}$ bit operations, 2.8430× smaller than the cost of collision decoding and 1.6488× smaller than the cost of collision decoding with the birthday trick.

Ball-collision decoding with parameters $k_1 = k_2 = 11484$, $\ell_1 = \ell_2 = 156$, $p_1 = p_2 = 29$, and $q_1 = q_2 = 1$ needs only $2^{996.21534}$ bit operations. This is 26.767× smaller than the cost of collision decoding, 15.523× smaller than the cost of collision decoding with the birthday trick, and 9.415× smaller than the Finiasz–Sendrier lower bound.

7 Asymptotic complexity of ball-collision decoding

This section analyzes the asymptotic behavior of the cost of ball-collision decoding, and shows that it always has a smaller asymptotic exponent than the cost of collision decoding.

Input sizes. Fix a real number W with $0 < W < 1/2$, and fix a real number R with $-W\log_2 W - (1-W)\log_2(1-W) \leq 1 - R < 1$.

Consider codes and error vectors of very large length n, where the codes have dimension $k \approx Rn$, and the error vectors have weight $w \approx Wn$. More precisely, fix functions $k, w : \{1, 2, \ldots\} \to \{1, 2, \ldots\}$ that satisfy $\lim_{n\to\infty} k(n)/n = R$ and $\lim_{n\to\infty} w(n)/n = W$; more concisely, $k/n \to R$ and $w/n \to W$.

Attack parameters. Fix real numbers P, Q, L with $0 \leq P \leq R/2$, $0 \leq Q \leq L$, and $0 \leq W - 2P - 2Q \leq 1 - R - 2L$. Fix ball-collision parameters $p_1, p_2, q_1, q_2, k_1, k_2, \ell_1, \ell_2$ with $p_i/n \to P$, $q_i/n \to Q$, $k_i/n \to R/2$, and $\ell_i/n \to L$.

We have also analyzed more general asymptotic parameter spaces, for example splitting P into P_1, P_2 where $p_i/n \to P_i$. Balanced parameters always turned out to be asymptotically optimal (as one would expect), so this section focuses on the parameter space (P, Q, L) stated above. Note that the asymptotic optimality of $P_1 = P_2$ does *not* imply the concrete optimality of $p_1 = p_2$; for example, $(p_1, p_2) = (2, 1)$ appears to be optimal for some small input sizes.

In the formulas below, expressions of the form $x\log_2 x$ are extended (continuously but not differentiably) to 0 at $x = 0$. For example, the expression $P\log_2 P$ means 0 if $P = 0$.

Success probability. We repeatedly invoke the standard asymptotic formula for binomial coefficients, namely

$$\frac{1}{n}\log_2 \binom{(\alpha+o(1))n}{(\beta+o(1))n} \to \alpha\log_2\alpha - \beta\log_2\beta - (\alpha-\beta)\log_2(\alpha-\beta),$$

to compute the asymptotic exponent of the success probability of a single iteration of ball-collision decoding:

$$\begin{aligned} B(P,Q,L) &= \lim_{n\to\infty}\frac{1}{n}\log_2\left(\binom{n}{w}^{-1}\binom{n-k-\ell_1-\ell_2}{w-p_1-p_2-q_1-q_2}\binom{k_1}{p_1}\binom{k_2}{p_2}\binom{\ell_1}{q_1}\binom{\ell_2}{q_2}\right) \\ &= W\log_2 W + (1-W)\log_2(1-W) \\ &\quad + (1-R-2L)\log_2(1-R-2L) - (W-2P-2Q)\log_2(W-2P-2Q) \\ &\quad - (1-R-2L-(W-2P-2Q))\log_2(1-R-2L-(W-2P-2Q)) \\ &\quad + R\log_2(R/2) - 2P\log_2 P - (R-2P)\log_2(R/2-P) \\ &\quad + 2L\log_2 L - 2Q\log_2 Q - 2(L-Q)\log_2(L-Q). \end{aligned}$$

The success probability of a single iteration is asymptotically $2^{n(B(P,Q,L)+o(1))}$.

Iteration cost. We similarly compute the asymptotic exponent of the cost of an iteration:

$$\begin{aligned}
&C(P,Q,L)\\
&\quad= \lim_{n\to\infty} \frac{1}{n} \log_2 \left(\binom{k_1}{p_1}\binom{\ell_1}{q_1} + \binom{k_2}{p_2}\binom{\ell_2}{q_2} + \binom{k_1}{p_1}\binom{\ell_1}{q_1}\binom{k_2}{p_2}\binom{\ell_2}{q_2} 2^{-\ell_1-\ell_2} \right)\\
&\quad= \max\{(R/2)\log_2(R/2) - P\log_2 P - (R/2-P)\log_2(R/2-P)\\
&\qquad\qquad + L\log_2 L - Q\log_2 Q - (L-Q)\log_2(L-Q),\\
&\qquad\qquad R\log_2(R/2) - 2P\log_2 P - (R-2P)\log_2(R/2-P)\\
&\qquad\qquad + 2L\log_2 L - 2Q\log_2 Q - 2(L-Q)\log_2(L-Q) - 2L\}.
\end{aligned}$$

The cost of a single iteration is asymptotically $2^{n(C(P,Q,L)+o(1))}$. Note that we have simplified the iteration cost to $\binom{k_1}{p_1}\binom{\ell_1}{q_1} + \binom{k_2}{p_2}\binom{\ell_2}{q_2} + \binom{k_1}{p_1}\binom{\ell_1}{q_1}\binom{k_2}{p_2}\binom{\ell_2}{q_2} 2^{-\ell_1-\ell_2}$. The cost is actually larger than this, but only by a factor $\le \mathrm{poly}(n)$, which we are free to disregard since $\frac{1}{n}\log_2 \mathrm{poly}(n) \to 0$. We also comment that the bounds are valid whether or not $q_i = 0$.

Overall attack cost. The overall asymptotic ball-collision-decoding-cost exponent is the difference $D(P,Q,L)$ of the iteration-cost exponent $C(P,Q,L)$ and the success-probability exponent $B(P,Q,L)$, thus

$$\begin{aligned}
D(P,Q,L) = \max\{&-(R/2)\log_2(R/2) + P\log_2 P + (R/2-P)\log_2(R/2-P)\\
&\quad - L\log_2 L + Q\log_2 Q + (L-Q)\log_2(L-Q), -2L\}\\
&- W\log_2 W - (1-W)\log_2(1-W)\\
&- (1-R-2L)\log_2(1-R-2L) + (W-2P-2Q)\log_2(W-2P-2Q)\\
&+ (1-R-2L-(W-2P-2Q))\log_2(1-R-2L-(W-2P-2Q)).
\end{aligned}$$

Example: Take $W = 0.04$ and $R = 1 + W\log_2 W + (1-W)\log_2(1-W) = 0.7577078109\ldots$. Choose $P = 0.004203556640625$, $Q = 0.000192998046875$, and $L = 0.017429431640625$; we use very high precision here to simplify verification. The success-probability exponent is $-0.0458435310\ldots$, and the iteration-cost exponent is $0.0348588632\ldots$, so the overall cost exponent is $0.0807023942\ldots$. Ball-collision decoding with these parameters thus costs $2^{(0.0807023942\ldots+o(1))n}$ to correct $(0.04+o(1))n$ errors in a code of rate $0.7577078109\ldots + o(1)$.

Collision-decoding cost and the lower bound. Traditional collision decoding is the special case $p_1 = p_2$, $k_1 = k_2$, $\ell_1 = \ell_2$, $q_1 = q_2 = 0$ of ball-collision decoding. Its asymptotic cost exponent is the case $Q = 0$ of the ball-collision decoding exponent stated above.

Consider again $W = 0.04$ and $R = 1 + W\log_2 W + (1-W)\log_2(1-W)$. Choosing $P = 0.00415087890625$, $Q = 0$, and $L = 0.0164931640625$ achieves decoding exponent $0.0809085120\ldots$. We partitioned the (P,L) space into small intervals and performed interval-arithmetic calculations to show that $Q = 0$

cannot do better than 0.0809; ball-collision decoding therefore has a slightly smaller exponent than collision decoding in this case.

We performed similar calculations for other pairs (W, R) and in each case found that the infimum of all collision-decoding-cost exponents was beaten by a ball-collision-decoding-cost exponent. Ball-collision decoding therefore has a smaller exponent than collision decoding, as stated in the introduction of this paper.

The case $Q = 0$ is always suboptimal. The interval-arithmetic calculations described above are proofs of the suboptimality of $Q = 0$ for some specific pairs (W, R). These proofs have the advantage of computing explicit bounds on the collision-decoding-cost exponents for those pairs (W, R), but the proofs have two obvious disadvantages.

The first disadvantage is that these proofs do not cover *all* pairs (W, R); they leave open the possibility that ball-collision decoding has the same exponent as collision decoding for other pairs (W, R). The second disadvantage is that the proofs are much too long to verify by hand. The first disadvantage could perhaps be addressed by much more extensive interval-arithmetic calculations, partitioning the space of pairs (W, R) into boxes so small that, within each box, the ball-collision-decoding exponent is uniformly better than the minimum collision-decoding exponent; but this would exacerbate the second disadvantage.

To address both of these disadvantages we give, in the full version of this paper [9], a hand-verifiable proof that $Q = 0$ is always suboptimal: for *every* (W, R), ball-collision decoding has a smaller asymptotic cost exponent than collision decoding. Specifically, we prove the following theorem about the overall asymptotic cost exponent:

Theorem 7.1 *For each R, W it holds that*

$$\begin{aligned}&\min\{D(P,0,L) : 0 \le P \le R/2,\ 0 \le W-2P \le 1-R-2L\}\\ >\ &\min\{D(P,Q,L) : 0 \le P \le R/2,\ 0 \le Q \le L,\ 0 \le W-2P-2Q \le 1-R-2L\}.\end{aligned}$$

Note that $\{(P, 0, L)\}$ and $\{(P, Q, L)\}$ are compact sets, and D is continuous, so we are justified in writing "min" rather than "inf". The proof strategy analyzes generic perturbations of D and combines all necessary calculations into a small number of elementary inequalities in the proofs in the full version of this paper [9].

8 Choosing McEliece parameters

The traditional approach to selecting cryptosystem parameters is as follows:

- Consider the fastest known attacks against the system. For example, in the case of RSA, consider the latest refinements [35] of the number-field sieve.
- Restrict attention to parameters for which these attacks take time at least $2^{b+\delta}$. Here b is the desired security level, and δ is a "security margin" meant to protect against the possibility of further improvements in the attacks.

– Within the remaining parameter space, choose the most efficient parameters. The definition of efficiency depends on the target application: it could mean minimal key size, for example, or minimum decryption time.

This approach does not make clear how to choose the security margin δ. Some applications have ample time and space for cryptography, and can simply increase δ to the maximum value for which the costs of cryptography are still insignificant; but in some applications cryptography is an important bottleneck, and users insist on minimizing δ for the sake of performance.

Finiasz and Sendrier in [28] identified a bound on "future improvements" in attacks against the McEliece cryptosystem, and suggested that designers use this bound to "choose durable parameters". The general idea of identifying bottlenecks in any possible attack, and of using those bottlenecks to systematically choose δ, is quite natural and attractive, and has been used successfully in many contexts. However, as discussed in Section 6, ball-collision decoding disproves the specific bound in [28], violating one of the assumptions in [28] and raising the question of how many more assumptions can be violated.

We propose replacing the bound in [28] with the simpler bound

$$\min\left\{\frac{1}{2}\binom{n}{w}\binom{n-k}{w-p}^{-1}\binom{k}{p}^{-1/2} : p \geq 0\right\};$$

i.e., choosing the code length n, code rate k/n, and error fraction w/n so that this bound is at least 2^b. As usual, implementors can exploit the remaining flexibility in parameters to optimize decryption time, compressed key size $k(n-k)$, or efficiency in any other metric of interest.

This bound has several attractive features. It is easy to estimate via standard binomial-coefficient approximations. It is easy to compute exactly. It covers a very wide class of attacks, as explained in the full version [9] of this paper. It is nevertheless in the same ballpark as the cost of known attacks: for example, it is $2^{49.69}$ for the original parameters $(n,k,w) = (1024, 524, 50)$, and $2^{236.49}$ for $(n,k,w) = (6624, 5129, 117)$. Note that these numbers give lower bounds on the cost of the attack. Parameters protecting against this bound pay only about a 20% performance penalty at high security levels, compared to parameters that merely protect against known attacks.

The reader can easily verify that parameters $(n,k,w) = (3178, 2384, 68)$ achieve 128-bit security against this bound. For 256-bit security $(n,k,w) = (6944, 5208, 136)$ are recommended.

References

[1] C.M. Adams, H. Meijer, *Security-related comments regarding McEliece's public-key cryptosystem*, in Crypto '87 [46] (1987), 224–228; see also newer version [2]. Citations in this document: §4.

[2] C.M. Adams, H. Meijer, *Security-related comments regarding McEliece's public-key cryptosystem*, IEEE Transactions on Information Theory **35** (1988), 454–455; see also older version [1]. Citations in this document: §1, §4.

[3] A. Al Jabri, *A statistical decoding algorithm for general linear block codes*, in IMA 2001 [31] (2001), 1–8. Citations in this document: §4.

[4] A.E. Ashikhmin, A. Barg, *Minimal vectors in linear codes*, IEEE Transactions on Information Theory **44** (1998), 2010–2017. Citations in this document: §4.

[5] A. Barg, E.A. Krouk, H.C.A. van Tilborg, *On the complexity of minimum distance decoding of long linear codes*, IEEE Transactions on Information Theory **45** (1999), 1392–1405. Citations in this document: §4, §4, §4, §4, §4, §4.

[6] L. Batten, R. Safavi-Naini (editors), *Information security and privacy: 11th Australasian conference, ACISP 2006, Melbourne, Australia, July 35, 2006, proceedings*, Lecture Notes in Computer Science, 4058, Springer, 2006. See [43].

[7] D.J. Bernstein, J. Buchmann, E. Dahmen (editors), *Post-quantum cryptography*, Springer, 2009. See [44].

[8] D.J. Bernstein, T. Lange, C. Peters, *Attacking and defending the McEliece cryptosystem*, in PQCrypto 2008 [14] (2008), 31–46. URL: http://eprint.iacr.org/2008/318. Citations in this document: §1, §1, §3, §3, §4, §4, §6.

[9] D.J. Bernstein, T. Lange, C. Peters, *Smaller decoding exponents: ball-collision decoding (full version)* (2010). URL: http://eprint.iacr.org/2010/585. Citations in this document: §7, §7, §8.

[10] D.J. Bernstein, T. Lange, C. Peters, H.C.A. van Tilborg, *Explicit bounds for generic decoding algorithms for code-based cryptography*, in WCC 2009 (2009). Citations in this document: §5.

[11] T.A. Berson, *Failure of the McEliece public-key cryptosystem under message-resend and related-message attack*, in Crypto '97 [33] (1997), 213–220. Citations in this document: §1.

[12] M. Blaum, P.G. Farrell, H.C.A. van Tilborg (editors), *Information, coding and mathematics*, Kluwer International Series in Engineering and Computer Science, 687, Kluwer, 2002. See [53].

[13] R.P. Brent, H.T. Kung, *The area-time complexity of binary multiplication*, Journal of the ACM **28** (1981), 521–534. URL: http://wwwmaths.anu.edu.au/~brent/pub/pub055.html. Citations in this document: §6.

[14] J. Buchmann, J. Ding (editors), *Post-quantum cryptography, second international workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, proceedings*, Lecture Notes in Computer Science, 5299, Springer, 2008. See [8].

[15] P. Camion, P. Charpin, S. Harari (editors), *Eurocode '92: proceedings of the international symposium on coding theory and applications held in Udine, October 23–30, 1992*, Springer, 1993. See [20].

[16] A. Canteaut, H. Chabanne, *A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem*, in EUROCODE 94 [21] (1994). URL: http://www.inria.fr/rrrt/rr-2227.html. Citations in this document: §4.

[17] 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 Transactions on Information Theory **44** (1998), 367–378. URL: ftp://ftp.inria.fr/INRIA/tech-reports/RR/RR-2685.ps.gz. Citations in this document: §3, §4.

[18] A. Canteaut, N. Sendrier, *Cryptanalysis of the original McEliece cryptosystem*, in Asiacrypt '98 [42] (1998), 187–199. Citations in this document: §3, §4.

[19] H. Chabanne, B. Courteau, *Application de la méthode de décodage itérative d'Omura à la cryptanalyse du système de McEliece*, Université de Sherbrooke, Rapport de Recherche, number 122 (1993). Citations in this document: §4.

[20] F. Chabaud, *Asymptotic analysis of probabilistic algorithms for finding short codewords*, in [15] (1993), 175–183. Citations in this document: §4.

[21] P. Charpin (editor), *Livre des résumé — EUROCODE 94, Abbaye de la Bussière sur Ouche, France, October 1994*, 1994. See [16].

[22] G.C. Clark, Jr., J. Bibb Cain, *Error-correcting coding for digital communication*, Plenum, 1981. Citations in this document: §4.

[23] J.T. Coffey, R.M. Goodman, *The complexity of information set decoding*, IEEE Transactions on Information Theory **35** (1990), 1031–1037. Citations in this document: §4.

[24] J.T. Coffey, R.M. Goodman, P. Farrell, *New approaches to reduced complexity decoding*, Discrete and Applied Mathematics **33** (1991), 43–60. Citations in this document: §4, §5.

[25] G.D. Cohen, J. Wolfmann (editors), *Coding theory and applications*, Lecture Notes in Computer Science, 388, Springer, 1989. See [50].

[26] I.I. Dumer, *Two decoding algorithms for linear codes*, Problemy Peredachi Informatsii **25** (1989), 24–32. Citations in this document: §4.

[27] I.I. Dumer, *On minimum distance decoding of linear codes*, in [32] (1991), 50–52. Citations in this document: §4.

[28] M. Finiasz, N. Sendrier, *Security bounds for the design of code-based cryptosystems*, in Asiacrypt 2009 [40] (2009). URL: http://eprint.iacr.org/2009/414. Citations in this document: §1, §4, §4, §4, §4, §6, §2, §8, §8, §8, §8.

[29] S. Goldwasser (editor), *Advances in cryptology — CRYPTO '88, proceedings of the conference on the theory and application of cryptography held at the University of California, Santa Barbara, California, August 21–25, 1988*, Lecture Notes in Computer Science, 403, Springer, 1990. See [51].

[30] C.G. Günther, *Advances in cryptology — EUROCRYPT '88, proceedings of the workshop on the theory and application of cryptographic techniques held in Davos, May 25–27, 1988*, Lecture Notes in Computer Science, 330, Springer-Verlag, Berlin, 1988. See [38].

[31] B. Honary (editor), *Cryptography and coding: proceedings of the 8th IMA international conference held in Cirencester, December 17–19, 2001*, Lecture Notes in Computer Science, 2260, Springer, 2001. See [3].

[32] G.A. Kabatianskii (editor), *Fifth joint Soviet-Swedish international workshop on information theory, Moscow, 1991*, 1991. See [27].

[33] B.S. Kaliski Jr. (editor), *Advances in cryptology — CRYPTO '97: 17th annual international cryptology conference, Santa Barbara, California, USA, August 17–21, 1997, proceedings*, Lecture Notes in Computer Science, 1294, Springer, 1997. See [11].

[34] K. Kim (editor), *Public key cryptography: proceedings of the 4th international workshop on practice and theory in public key cryptosystems (PKC 2001) held on Cheju Island, February 13–15, 2001*, Lecture Notes in Computer Science, 1992, Springer, 2001. See [36].

[35] T. Kleinjung, K. Aoki, J. Franke, A.K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P.L. Montgomery, D.A. Osvik, H. te Riele, A. Timofeev, P. Zimmermann, *Factorization of a 768-bit RSA modulus*, in Crypto 2010 [48] (2010), 333–350. URL: http://eprint.iacr.org/2010/006. Citations in this document: §8.

[36] K. Kobara, H. Imai, *Semantically secure McEliece public-key cryptosystems — conversions for McEliece PKC*, in PKC 2001 [34] (2001), 19–35. Citations in this document: §1.

[37] E.A. Krouk, *Decoding complexity bound for linear block codes*, Problemy Peredachi Informatsii **25** (1989), 103–107. Citations in this document: §4, §4.

[38] P.J. Lee, E.F. Brickell, *An observation on the security of McEliece's public-key cryptosystem*, in Eurocrypt '88 [30] (1988), 275–280. URL: http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E88/275.PDF. Citations in this document: §4.

[39] J.S. Leon, *A probabilistic algorithm for computing minimum weights of large error-correcting codes*, IEEE Transactions on Information Theory **34** (1988), 1354–1359. Citations in this document: §4.

[40] M. Matsui (editor), *Advances in cryptology — ASIACRYPT 2009, 15th international conference on the theory and application of cryptology and information security, Tokyo, Japan, December 6–10, 2009, proceedings*, Lecture Notes in Computer Science, 5912, Springer, 2009. See [28].

[41] R.J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, JPL DSN Progress Report (1978), 114–116. URL: http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Citations in this document: §1, §4.

[42] K. Ohta, D. Pei (editors), *Advances in cryptology — ASIACRYPT'98: proceedings of the international conference on the theory and application of cryptology and information security held in Beijing*, Lecture Notes in Computer Science, 1514, Springer, 1998. See [18].

[43] R. Overbeck, *Statistical decoding revisited*, in ACISP 2006 [6] (2006), 283–294. Citations in this document: §4.

[44] R. Overbeck, N. Sendrier, *Code-based cryptography*, in [7] (2009), 95–145. Citations in this document: §2, §4.

[45] C. Peters, *Information-set decoding for linear codes over* $\mathbf{F}_q$, in Post-Quantum Cryptography [49] (2010), 81–94. Citations in this document: §1, §4, §2.

[46] C. Pomerance (editor), *Advances in cryptology — CRYPTO '87, proceedings of the conference on the theory and applications of cryptographic techniques held at the University of California, Santa Barbara, California, August 16–20, 1987*, Lecture Notes in Computer Science, 293, Springer, 1987. URL: http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C87/224.PDF. See [1].

[47] E. Prange, *The use of information sets in decoding cyclic codes*, IRE Transactions on Information Theory **IT-8** (1962), S5–S9. Citations in this document: §4.

[48] T. Rabin (editor), *Advances in cryptology — CRYPTO 2010, 30th annual cryptology conference, Santa Barbara, CA, USA, August 15–19, 2010, proceedings*, Lecture Notes in Computer Science, 6223, Springer, 2010. See [35].

[49] N. Sendrier (editor), *Post-quantum cryptography, third international workshop, PQCrypto, Darmstadt, Germany, May 25-28, 2010, proceedings*, Lecture Notes in Computer Science, 6061, Springer, 2010. See [45].

[50] J. Stern, *A method for finding codewords of small weight*, in [25] (1989), 106–113. Citations in this document: §1, §3, §3, §4, §4.

[51] J. van Tilburg, *On the McEliece public-key cryptosystem*, in Crypto '88 [29] (1990), 119–131. Citations in this document: §4.

[52] J. van Tilburg, *Security-analysis of a class of cryptosystems based on linear error-correcting codes*, Ph.D. thesis, Technische Universiteit Eindhoven, 1994. Citations in this document: §4.

[53] E.R. Verheul, J.M. Doumen, H.C.A. van Tilborg, *Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece public-key cryptosystem*, in [12] (2002), 99–119. Citations in this document: §1.