On lower bounds for information set decoding over $\mathbb{F}_q$ and on the effect of partial knowledge

Robert Niebuhr

Artilleriestrasse 11,
Darmstadt 64285, Merck KGaA, Germany
Email: robert.niebuhr@gmx.net

Edoardo Persichetti*

Dakota State University,
College of Arts and Sciences,
820 N Washington Ave,
Madison, SD 57042, USA
Email: edoardo.persichetti@dsu.edu
*Corresponding author

Pierre-Louis Cayrel

Laboratoire Hubert Curien,
Université Jean Monnet,
UMR CNRS 5516, Bâtiment F 18 Rue du Professeur Benoît Lauras,
Saint-Etienne 42000, France
Email: pierre.louis.cayrel@univ-st-etienne.fr

Stanislav Bulygin

Germany
Email: bulygin5@googlemail.com

Johannes Buchmann

Technische Universität Darmstadt,
Fachbereich Informatik Kryptographie und Computeralgebra,
Hochschulstrasse 10,
Darmstadt 64289, Germany
Email: buchmann@cdc.informatik.tu-darmstadt.de

Abstract: Code-based cryptosystems are promising candidates for post-quantum cryptography since they are fast, require only basic arithmetic because their security is well understood. The increasing number of cryptographic schemes based on codes over fields other than $\mathbb{F}_2$ presents, however, security issues that are not relevant in the case of binary codes; the security of such constructions, therefore, requires separate assessment. Information set decoding (ISD) is one of the most important generic attacks against code-based cryptosystems. We give lower bounds for ISD over $\mathbb{F}_q$, thereby anticipating future software and

hardware improvements. Our results allow to compute conservative parameters for cryptographic applications. While most security proofs assume that an attacker does not have any additional information about the secret, we show that in certain scenarios an attacker can gain partial knowledge of the secret. We present how this knowledge can be used to improve the efficiency of an attack and give new bounds for the complexity of such an attack. In this paper, we analyse two types of partial knowledge including concrete scenarios and give an idea how to prevent the leakage of such knowledge to an attacker.

Keywords: codes; cryptography; information set decoding; lower bounds; partial knowledge; post-quantum.

Reference to this paper should be made as follows: Niebuhr, R., Persichetti, E., Cayrel, P-L., Bulygin, S. and Buchmann, J. (2017) 'On lower bounds for information set decoding over $\mathbb{F}_q$ and on the effect of partial knowledge', *Int. J. Information and Coding Theory*, Vol. 4, No. 1, pp.47–78.

Biographical notes: Robert Niebuhr was born in Offenbach a.m. in 1981. He studied Mathematics at the University of Technology, Darmstadt, and University of New South Wales, Sydney. After finishing his Diploma thesis on Algebraic-Geometric Codes in 2006, he worked for 2 years as a Management Consultant. In 2008, he took a leave of absence and returned to Darmstadt to work on his PhD thesis under the supervision of Professor J. Buchmann and Dr. P-L. Cayrel.

Edoardo Persichetti was born in Rome, Italy in 1984. He studied Mathematics at University 'Sapienza' in Rome for his Bachelor's and Master's. He then moved to New Zealand in 2009 to study for his PhD in Cryptography under the supervision of Steven Galbraith. In 2012, he moved to Poland for a 2-year Postdoc at University of Warsaw, in the group directed by Stefan Dziembowski. He is currently an Assistant Professor at Dakota State University.

Pierre-Louis Cayrel was born in France. He studied Mathematics at the University of Avignon until he obtained his Master's. Then he went to Limoges to begin a PhD in Code-Based Cryptography under the supervision of Professor P. Gaborit. He also worked on algebraic attacks against stream ciphers. In 2008-2009, he was an ATER in Paris 8 in the group of Professor C. Carlet. Since September 2009, Dr. Cayrel is working as a Post Doc in the Center of Advanced Research Security Darmstadt (CASED) research area of Secure Data, he is in charge of co-supervising (with Prof. Buchmann) three PhD students working on code-based cryptography.

Stanislav Bulygin was born in Kiev, Ukraine in 1982. He studied Mathematics at the National Kiev University and at the University of Kaiserslautern (Germany). He got his Diploma from the former one in 2005 and from the latter one in 2006. In 2006–2009, he was a PhD Candidate at the University of Kaiserslautern under the supervision of Professor Dr. G-M. Greuel and received his PhD in June 2009. In the period of 2009–2013, he worked at the Technische Universität Darmstadt (Germany) with a research focus on cryptography (in particular post-quantum cryptography), cryptanalysis and coding theory. Since 2013, he has worked at different companies in the fields of IT-security, mobile payment and e-commerce.

Johannes A. Buchmann is a Professor of Computer Science and Mathematics at the Technische Universität Darmstadt. He received the most prestigious award in science in Germany, the Leibniz Award of the German Science Foundation. He also received the Karl Heinz-Beckurts Award for technology transfer. He is a member of the German Academy of Science and Engieneering.

1 Introduction

In Shor (1994) showed that quantum computers can break most 'classical' cryptosystems, e.g. those based on the integer factorisation problem or on the discrete logarithm problem. It is, therefore, crucial to develop cryptosystems that are resistant to quantum computer attacks. Cryptography based on error-correcting codes is a very promising candidate for post-quantum cryptography since code-based cryptographic schemes are usually fast and do not require special hardware, specifically no cryptographic co-processor.

Error-correcting codes have been applied in cryptography for at least three decades, ever since McEliece published his paper in McEliece (1978). The McEliece scheme is as old as RSA and has resisted cryptanalysis to date (except for a parameter adjustment). His work has received much attention as it is a promising candidate for post-quantum cryptography. McEliece used the class of binary Goppa codes for his construction, and most other schemes published since then have also been using binary codes.

However, in recent years, many new proposals use codes over larger fields $\mathbb{F}_q$, mostly in an attempt to reduce the size of the public and private keys. Two examples that received a lot of attention are quasi-cyclic (QC) Berger et al. (2009), and quasi-dyadic (QD) Misoczki and Barreto (2009); Persichetti (2012) codes. The security of these code-based constructions, however, is not as well understood for q-ary codes as for binary ones. For instance, several attacks on QC and QD codes have been published, indicating how to break some of the proposed non-binary parameter sets Faugère et al. (2010), Faugère et al. (2010), Umana and Leander (2009). Binary codes seem to be unaffected by these attacks.

The two most important types of attacks against code-based cryptosystems are structural attacks and decoding attacks. Structural attacks exploit structural weaknesses in the construction, and they attempt to recover the private key. For example, Faugère et al. (2010) uses the QC structure of the matrix to decrease the number of variables in the linear equation system used to recover the secret key. Decoding attacks are used to decrypt a given ciphertext. In this paper, we will focus on a particular type of decoding attacks, based on so-called information set decoding (ISD). Attacks based on ISD are among the most important generic decoding attacks, and they are the most efficient against several encryption and identification schemes.

To analyse the security of code-based schemes, cryptanalysts develop and improve (generic as well as specific) attacks, and propose lower bounds for the complexity of such attacks. All of these developments, however, assume that an attacker has no knowledge about the private key.

1.1 Partial knowledge

In some scenarios, an attacker can obtain partial knowledge of the private information and exploit this to improve the efficiency of an attack. Examples are:

1 **Schemes that use a restricted error vector domain**: Some cryptographic schemes, e.g. NTRU Hoffstein, Pipher and Silverman (1998), restrict the domain of an error vector. In this example, while the scheme itself is defined[1] over $\mathbb{F}_q$ for $q = 128$ or

$q = 256$, the error vector e is ternary, i.e. $e \in \{0, 1, -1\}^n$. ISD algorithms can be used to attack such systems, and we will show how to exploit the knowledge about the error vector domain. We analyse this example in more detail in Section 4.4.

2 **Schemes that leak information about the error vector entries**: In Stern's identification (ID) scheme Stern (1993), the prover sends a random permutation of the private vector to the verifier. This reveals the non-zero values of the vector, while their positions remain secret.This information is useless when binary codes are used (as in the original scheme), it *does* give the attacker an advantage when codes over $\mathbb{F}_q$ are used like in Cayrel, Véron and El Yousfi Alaoui (2010). We will analyse this example in more detail in Section 4.3.

1.2 Previous work

Many improvements and generalisations of ISD-like attacks have already been proposed in the literature; amongst these, we outline these pivotal examples:

1962 Prange (1962): The first proposal to use information sets for decoding

1988 Leon (1988): Prange's proposal is made probabilistic, thereby improving its runtime

1988 Lee and Brickell (1988): Introduced a systematic way to determine whether the recovered message is correct and proposed several other improvements. These include bitwise computation of vectors, which stops the computation earlier, and a speedup of the Gaussian elimination which reuses part of the columns of the parity-check matrix.

1990 van Tilburg (1988): The efficiency of the algorithm is further improved, and the ISD attack is made more efficient by a systematic method of checking and by using a random bit swapping procedure.

1989 Stern (1989): Proposed an idea that allows to make use of the birthday paradox, thereby increasing the speed of the search step.

1998 Canteaut and Chabaud (1998): This paper improved the efficiency of previous algorithms and used Markov chains to model the algorithm's behaviour.

2008 Bernstein, Lange and Peters (2008): Several techniques are proposed to further speed up the attack algorithm, e.g. by reusing pivot values to speed up the matrix inversion step.

2011 Bernstein, Lange and Peters (2011): In the so-called 'Ball collision' decoding algorithm, the weight of the error vector is fragmented further, leading to a speed-up.

2011 May, Meurer and Thomae (2011): The authors adapted the algorithm proposed by Howgrave-Graham and Joux for the subset sum problem Howgrave-Graham and Joux (2010).

2012 Becker et al. (2012): The previous variant is further refined thanks to two additional conditions, leading again to a small gain.

2015 May and Ozerov (2015): The authors adapted the algorithm in *High-Dimensional Nearest Neighbor* Har-Peled, Indyk and Motwani (2012), Andoni et al. (2014), Dubiner (2010).

Our work stems from two papers — by Finiasz and Sendrier (2009) and Peters (2010). The former provides lower bounds for the complexity of the ISD algorithm over $\mathbb{F}_2$, the latter describes how to generalise Stern's and Lee-Brickell's algorithms to $\mathbb{F}_q$. We are not aware of any previous work that specifically deals with the effects of partial knowledge.

1.3 Our contribution

In this paper, we propose and prove lower bounds for the complexity of ISD algorithms over a finite field $\mathbb{F}_q$. We generalise the lower bounds proposed by Finiasz and Sendrier (2009) to codes over $\mathbb{F}_q$ and fix a misapproximation in their proof (the approximation of the parameter l is suboptimal). In Bernstein, Lange and Peters (2010), the authors pointed out that the lower bound cannot be correct by showing how to decrease the algorithm complexity below the corresponding lower bound. We also use a more general attack algorithm to derive the lower bound that takes the improvement in Bernstein, Lange and Peters (2010) into account. This improvement is based on the fact that vectors with a single non-zero entry can be added using less operations than the addition of random vectors. While the authors of Bernstein, Lange and Peters (2010) also propose a new bound for the complexity of (binary) ISD, this bound is looser than ours. For the parameter set (12,200, 9,244, 488) mentioned in the paper, their bound yields approx. 890 bits of security, compared with 991 bits for our lower bound.

In addition to that, we show how to use the structure of $\mathbb{F}_q$ to increase the algorithm efficiency and compare our lower bounds with the ISD algorithm described by Peters (2010). The details of the proof are given in Appendix A.

Based on this generalisation, we analyse two types of partial knowledge an attacker can obtain in certain scenarios. We show that additional knowledge can be used to improve the efficiency of an attack by restricting the space that needs to be searched, and we prove new lower bounds for these cases. As a first example, we analyse the Cayrel-Véron-El Yousfi (CVE) ID scheme Cayrel, Véron and El Yousfi Alaoui (2010) to apply our methodology and assess their method to prevent the information leak. A second example shows how to attack the NTRU scheme using our modified ISD algorithm.

This work is an extension of two conference papers: the generalisation of ISD to $\mathbb{F}_q$ was presented in Niebuhr et al. (2010b) (without formal proceedings), and the impact of partial knowledge was analysed in Niebuhr et al. (2010a).[2] The additional contribution of this merged version is to provide a self-contained paper that includes all necessary proofs. The new section on CVE provides a better example on possible information leaks and how to prevent them. Also, this merged version allows to view both works in context, since the two parts are strongly linked to one another.

1.4 Organisation of the paper

In Section 2, we start with an overview of coding theory and code-based cryptography over $\mathbb{F}_q$. The subsequent Section 3 presents the idealised ISD algorithm we are analysing and states the lower bound result. We apply these lower bounds to concrete parameters and compare the results with the most recent algorithm. Section 4 covers the analysis of

the mentioned types of partial knowledge. In the following, we apply our results to the example of the CVE ID scheme and to NTRU. We conclude in Section 5.

2 Overview of code-based cryptography

2.1 Coding theory over $\mathbb{F}_q$

A linear code $\mathcal{C}$ is a k-dimensional subspace of an n-dimensional vector space over a finite field $\mathbb{F}_q$. The code parameters k and n are positive integers with $k \leq n$, q is a prime power, and a code with these parameters is denoted $[n, k]$-code.

Definition 1 (Hamming weight)**:** The (Hamming) weight $\mathrm{wt}(x)$ of a vector x is the number of its non-zero entries.

Definition 2 (Minimum distance)**:** The (Hamming) distance $d(x, y)$ between two vectors $x, y \in \mathbb{F}_q^n$ is defined as the (Hamming) weight of $x - y$. The minimum weight d of a code $\mathcal{C}$ is defined as the minimum distance between any two different codewords, or equivalently as the minimum weight over all non-zero codewords:

$$d := \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d(x, y) = \min_{\substack{c \in \mathcal{C} \\ c \neq 0}} \mathrm{wt}(c).$$

A linear code of length n, dimension k, and minimum distance d is called an $[n, k, d]$-code.

The error-correcting capability of such a code equals $\lfloor (d-1)/2 \rfloor$ (under bounded-distance decoding). By (n, k, t), we denote a code that can efficiently decode up to $t \leq \lfloor (d-1)/2 \rfloor$ errors. The co-dimension r of this code is defined as $r := n - k$.

Definition 3 (Generator and parity check matrix)**:** Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$. A generator matrix G of $\mathcal{C}$ is a matrix whose rows form a basis of $\mathcal{C}$:

$$\mathcal{C} = \{xG : x \in \mathbb{F}_q^k\}.$$

A parity check matrix H of $\mathcal{C}$ is defined by the characterising property

$$\mathcal{C} = \{x \in \mathbb{F}_q^n : Hx^T = 0\}$$

and it generates the dual space of $\mathcal{C}$. For a given parity check matrix H and any vector e, we call s the *syndrome* of e where $s^T := He^T$.

Remark 1: Generator and parity check matrices are always of full rank.

Two generator matrices generate *equivalent codes* if one is obtained from the other by a linear transformation or permutation of columns. Therefore, we can write any generator matrix G in *systematic form* $G = [I_k|R]$, where I_k is the identity matrix of size k and $|$ denotes concatenation. This allows a more compact representation of the matrix. If $\mathcal{C}$ is generated by $G = [I_k|R]$, then a parity check matrix for $\mathcal{C}$ is $H = [-R^T|I_{n-k}]$ (up to permutation, H can be transformed, so that the identity submatrix is on the left-hand side).

For any generator matrix G and parity check matrix H of a code, we have $GH^T = 0$.

Remark 2: For an $[n, k, d]$-code, the generator matrix G is of size $k \times n$, and the parity check matrix has size $(n-k) \times n = r \times n$.

The Gilbert-Varshamov (GV) bound provides a bound on the maximum size of a code and states the existence of codes with certain parameters.

Theorem 1 (q-ary Gilbert-Varshamov bound)**:** *Let $A_q(n, d)$ be the maximum size (maximum number of codewords) of a q-ary code of length n and minimum distance d. Then:*

$$A_q(n,d) \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j}(q-1)^j}.$$

Let n, k and d_0 be positive integers such that

$$\sum_{i=0}^{d_0-2} \binom{n-1}{i}(q-1)^i < q^r,$$

with $r = n - k$. Then there exists an $[n, k, d_0]$ code.

Proof: See Varshamov (1957). □

Remark 3: Since for large n, the last term dominates the sum, the bound is often approximated by

$$\binom{n}{d_0-2}(q-1)^{d_0-2} < q^r.$$

Random codes, which are used in the Stern ID scheme and the CVE ID scheme below, satisfy the GV bound with equality.

2.2 The syndrome decoding problem and the McEliece PKC

Problem 1: *Given an $(n-k) \times n$ full-rank matrix H and a vector s, both with entries in $\mathbb{F}_q$, and a non-negative integer t; find a vector $e \in \mathbb{F}_q^n$ of weight t such that $He^T = s^T$.*

The corresponding decision problem was proved to be NP-complete in Berlekamp, McEliece and van Tilborg (1978), but only for binary codes. In 1994, A. Barg proved that this result holds for codes over all finite fields [(Barg, 1994, in Russian) and (Barg, 1997, Theorem 4.1)].

Many code-based cryptographic constructions are based on the hardness of the syndrome decoding problem. Among those are the McEliece encryption scheme McEliece (1978) and the CFS signature scheme Courtois, Finiasz and Sendrier (2001). The latter, however, is unsuitable for q-ary codes, since it requires codes with high density of decodable words, and the density rapidly decreases with an increasing field size q. We will, therefore, briefly describe the McEliece encryption scheme and show how it can be attacked by solving the syndrome decoding problem.

2.2.1 The McEliece PKC

The McEliece public-key encryption scheme was presented by McEliece (1978). The original scheme uses binary Goppa codes, for which it remains unbroken, but the scheme can be used with any class of codes for which an efficient decoding algorithm is known.

Let G' be a generator matrix for a linear (n, k, t)-code over $\mathbb{F}_q$, and $\mathcal{D}_{G'}$ be a corresponding efficient decoding algorithm that can correct up to t errors. Let P be a random $n \times n$ permutation matrix and S an invertible $k \times k$ matrix over $\mathbb{F}_q$. These form the private key, while (G, t) is made public, where $G := SG'P$.

Encryption: Represent the plaintext as a vector m of length k over $\mathbb{F}_q$, choose a q-ary random error vector e of weight t, and compute the ciphertext

$$c = mG + e.$$

Decryption: Compute

$$\widehat{c} = cP^{-1} = mSG' + eP^{-1}.$$

As P is a permutation matrix, eP^{-1} has the same weight as e. Therefore, $\mathcal{D}_{G'}$ can be used to decode these errors:

$$\hat{m} = mS = \mathcal{D}_{G'}(\widehat{c})$$

Thus, we can compute the plaintext

$$m = \hat{m}S^{-1}.$$

Remark 4: In the McEliece cryptosystem using an $[n, k, d]$ code, the number of error vectors is $\binom{n}{t}(q-1)^t$, while the number of possible syndromes is q^r. Therefore, if

$$\binom{n}{t}(q-1)^t < q^r,$$

there is at most one solution to the SD problem, so the decoding process is unique. Otherwise, there can be several solutions.

2.3 The birthday paradox

The algorithms in this paper make use of the famous birthday paradox. The name is due to the fact that a surprisingly small number of people are required such that the probability of at least two of them having the same birthday is greater than 50% (this number is 23).

In our context, the birthday paradox is used in the following way. In order to find p columns of a matrix H that sum up to a given vector s, we build two lists. L_1 contains sums of $p/2$ columns of H, and $L_2 = \{s - x : x \in L_1\}$. Finding a common entry between these lists corresponds to the birthday situation, hence we can expect to quickly find a large number of these (so-called) collisions.

2.4 Information set decoding

Information set decoding algorithms are among the most efficient generic attacks against code-based cryptosystems like the McEliece encryption scheme, the CFS signature scheme Courtois, Finiasz and Sendrier (2001), the stands for Fast Syndrome-Based (FSB) hash function Augot, Finiasz and Sendrier (2005), and others. Over the years, there have been many improvements and generalisations of ISD-like attacks.

ISD algorithms solve the decoding problem, i.e. the decoding code words with errors. More specifically, if m is a plaintext and $c = mG + e$ is a ciphertext, where e is a vector of weight t, then ISD algorithms take c as input and recover m (or, equivalently, e). Since $s^T := Hc^T = H(mG + e)^T = He^T$, the problem is often formulated using a parity check matrix as in Problem 1.

If the number of errors that have to be corrected is smaller than the GV bound, then there is at most one solution. Otherwise, there can be several solutions.

A basic version of an ISD algorithm works as follows: a random permutation P is applied to H in the hope that all columns corresponding to error positions in e are moved to the left-hand side of the matrix (in the first $n - k$ columns). Then Gaussian elimination is used to transform H into the form $H' = [I_{n-k}|R]$, where I_{n-k} is the $(n - k) \times (n - k)$ identity matrix, and the row operations are performed on s as well to get s'. If s' has a weight not exceeding t, the algorithm succeeds; we can read off the error positions from s' and get $e = P^{-1}[s'|0^k]$, where 0^k is the zero vector of size k. Otherwise, the algorithm restarts.

Most advanced ISD versions use of the birthday paradox: they allow a certain (usually small) number p of errors in the last $k - l$ columns of H. Then lists of column sums of H are used to find these error positions. If we split the right-hand part of H into $[H_1|H_2]^T$, and write $e = [e_1|e_2]$, then we search for a vector e_2 of weight p such that $s^T - H_2e_2^T$ has weight $t - p$, and the non-zero positions of $s^T - H_2e_2^T$ show the remaining $t - p$ error positions.

This idea is further refined in some papers, such as Bernstein, Lange and Peters (2011), where the weight distribution is characterised by an additional parameter q.

The next section describes the algorithm which is used for our analysis in more detail.

3 Lower bounds for information set decoding over $\mathbb{F}_q$

The security analysis of cryptosystems considers the most efficient known algorithm breaking the security of a cryptographic scheme and then it estimates the attack's complexity. However, when a faster attack is developed, this security analysis might become obsolete. Over the years, there have been many improvements for ISD. The authors of Finiasz and Sendrier (2009) proposed to establish lower bounds for an ISD family of algorithms to allow for conservative, longer lasting security estimates. Our algorithm is a generalisation of Finiasz and Sendrier (2009) to code over $\mathbb{F}_q$. This algorithm represents an idealised ISD-like algorithm, meaning that we only consider a cost for the most crucial steps and assume no cost for overhead, memory access etc. The most significant steps in ISD are the calculation of syndromes (ISD 1 and ISD 2 in Algorithm 1 on page 58) and checking the success condition (ISD 3).

We first describe how to modify the ISD algorithm to work over $\mathbb{F}_q$; then we show how to use the field structure to increase efficiency by a factor of $\sqrt{q-1}$.

3.1 Analysis of the algorithm

The algorithm we describe here recovers a q-ary error vector. In each step, we randomly rearrange the columns of the parity check matrix H and transform it into the form

$$H = \left(\begin{array}{c|c|c} I_{n-k-l} & 0 & H_1 \\ \hline 0 & I_l & H_2 \end{array}\right), \tag{1}$$

where I_{n-k-l} and I_l are identity matrices of size $(n-k-l)$ and size l, respectively. The columns are usually chosen adaptively to guarantee the success of this step, i.e., to ensure that $n-k$ linearly independent columns are found. Although this approach could bias the following steps, it has not shown any influence in practice. The variables l and p (see next step) are algorithm parameters optimised for each parameter set (n, k, t).

The error vector we are looking for has p_1 errors in the column set corresponding to H_1 and H_2, p_2 errors in the columns corresponding to I_l, and the remaining $(t-p_1-p_2)$ errors in the first $(n-k-l)$ columns. The reason for this split into p_1 and p_2 errors is that columns in the middle block of H only have one non-zero entry, which makes it faster to add them to another vector. We will go into more detail at the end of this section.

To find an error vector with the structure described above, many possible error patterns of $p := p_1 + p_2$ errors in the last $k+l$ columns are checked, such that the weighted sum $\hat{s}$ of those p columns is identical to the syndrome s in the last l entries. This is done by searching for collisions between two sets L_1 and L_2 defined as

$$L_1 = \{\hat{H}_2 e^T : e \in W_1\}, \tag{2}$$

$$L_2 = \{s_2 - \hat{H}_2 e^T : e \in W_2\}, \tag{3}$$

where $\hat{H}_2 = [I_l|H_2]$, $W_1 \subseteq \mathcal{W}_{k+l;\lfloor p/2 \rfloor;q}$ and $W_2 \subseteq \mathcal{W}_{k+l;\lceil p/2 \rceil;q}$ are given to the algorithm, and $\mathcal{W}_{k+l;p;q}$ is the set of all q-ary words of length $k+l$ and weight p such that the first l entries contain exactly p_2 errors. This approach corresponds to the *overlapping sets* technique proposed in Finiasz and Sendrier (2009). Alternatively, the sets can be defined such that they overlap only partially or not at all, but this has only a small influence in practice.

Writing $e = [e'|(e_1 + e_2)]$ and $s = [s_1|s_2]$ with s_2 of length l, we now search for vectors e_1 and e_2 of weight $\lfloor p/2 \rfloor$ and $\lceil p/2 \rceil$, respectively, such that

$$\hat{H}_2[e_1 + e_2]^T = s_2^T.$$

If this succeeds, we compute the difference $\hat{s} - s$; if this does not have weight $t-p$, the algorithm restarts. Otherwise, we write $\hat{H}_1 = [0|H_1]$, and then the non-zero entries correspond to the remaining $t-p$ errors:

$$\begin{aligned} He^T &= \left(\begin{array}{c|c|c} I_{n-k-l} & 0 & H_1 \\ \hline 0 & I_l & H_2 \end{array}\right) \begin{pmatrix} e'^T \\ (e_1+e_2)^T \end{pmatrix} \\ &= \begin{pmatrix} I_{n-k-l} \cdot e'^T + \hat{H}_1 (e_1+e_2)^T \\ \hat{H}_2 (e_1+e_2)^T \end{pmatrix} \\ &= \begin{pmatrix} I_{n-k-l} \cdot e'^T \\ (0^l)^T \end{pmatrix} + \hat{s}^T \\ &= \begin{pmatrix} s_1^T \\ s_2^T \end{pmatrix}. \end{aligned}$$

Therefore, we have

$$I_{n-k-l} \cdot e'^T = s_1^T - \hat{H}_1(e_1 + e_2)^T,$$

revealing the remaining columns of e.

Definition 4: For any code parameters n, r, t and q, we denote by $\mathrm{WF}_{\mathrm{qISD}}(n, r, t, q)$ the minimum work factor (in number of operations) of Algorithm 1 applied to an $(n, k = n - r, t)$-code over $\mathbb{F}_q$, i.e., after optimising l, p_1, p_2, W_1 and W_2 for these parameters.

In the algorithm described in this paper, all computations are done over $\mathbb{F}_q$, so the complexity also depends on the implementation of q-ary arithmetic. A naïve implementation yields an additional factor of $\log_2(q)$ for addition and $\log_2^2(q)$ for multiplication. There are several techniques to improve this, e.g. by lifting to $\mathbb{Z}[x]$ (for large q) or by precomputing exp and log tables (for small q). Especially for small q, this allows to make q-ary arithmetic nearly as fast as binary, so in order to gain conservative estimates, we neglect this factor.

3.1.1 Using the field structure

We can use the field structure of $\mathbb{F}_q$ to increase the algorithm's efficiency. While this improvement is one of our own contributions (and has been described in Niebuhr et al. (2010b)), it has been found independently by Minder and Sinclair (2009).

Note that, for all vectors e such that $He^T = s^T$, there are $q - 1$ pairwise different vectors e' such that $He'^T = as^T$ for some $a \in \mathbb{F}_q \backslash \{0\}$, namely $e' = ae$. Clearly, if we find such an e', we can calculate e and thus solve the syndrome decoding problem. We can modify the algorithm to allow it to find the vectors e' as well, thereby increasing the fraction of error vectors that are (implicitly) tested in each iteration by a factor of $q - 1$ (see Appendix A for a detailed description).

Since this fraction is calculated using $|W_1| \cdot |W_2|$, we can also keep the fraction constant and decrease the size of the sets W_i by a factor of $\sqrt{q-1}$ each. As the work factor in each iteration of the algorithm is linear in $|W_1| + |W_2|$, this increases the algorithm's efficiency by a factor $\sqrt{q-1}$.

A simple way to decrease the size of the sets W_i is to redefine them. For any vector a over $\mathbb{F}_q$, denote its leading element, i.e., its first non-zero entry, by $\mathrm{le}(a) \in \mathbb{F}_q \backslash \{0\}$, and let

$$W_1' \subseteq \{e \in \mathcal{W}_{k+l;\lfloor p/2 \rfloor;q} : \mathrm{le}(e) = 1\}, \tag{4}$$

$$L_1' = \left\{ (\hat{H}_2 e^T)(\mathrm{le}(\hat{H}_2 e^T))^{-1} : e \in W_1' \right\}, \tag{5}$$

$$L_2' = \left\{ (s_2 - \hat{H}_2 e^T)(\mathrm{le}(s_2 - \hat{H}_2 e^T))^{-1} : e \in W_2 \right\}. \tag{6}$$

Remark 5: Note that even though the calculation of each vector is more costly due to the final division by the leading coefficient, this is by far offset by the smaller number of vectors that must be computed.

The algorithm thus works as shown in Algorithm 1. The description is the one from Finiasz and Sendrier (2009), modified to include our improvement described above.

Algorithm 1 Information Set Decoding over $\mathbb{F}_q$

Parameters:

- Code parameters: integers n, $r = n - k$ and t, and a finite field $\mathbb{F}_q$.
- Algorithm parameters: two integers $p > 0$ and $l > 0$, and two sets $W_1 \subseteq \{e \in \mathcal{W}_{k+l;\lfloor p/2 \rfloor;q} : \mathrm{le}(e) = 1\}$ and $W_2 \subseteq \mathcal{W}_{k+l;\lceil p/2 \rceil;q}$.

Remark: The function $h_l(x)$ returns the last l bits of the vector $x \in \mathbb{F}_q^n$. The variables $y := \mathrm{le}(\hat{H}_2 e_1^T)$ and $z := \mathrm{le}(s - \hat{H}_2 e_2^T)$ are notational shortcuts.

Input: Matrix $H_0 \in \mathbb{F}_q^{r \times n}$ and a vector $s_0 \in \mathbb{F}_q^r$.

Output: A pair (P, e), consisting of permutation P and vector e, which allow to compute the message m as described above.

Repeat (MAIN LOOP)
  $P \leftarrow$ random $n \times n$ permutation matrix
  $(H, U) \leftarrow$ PGElim$(H_0 P)$ //Gauss elimination as in (1)
  $s \leftarrow s_0 U^T$
  for all $e_1 \in W_1$
    $i \leftarrow h_l(\hat{H}_2 e_1^T / y)$ (ISD 1)
    write(e_1, i) //store e_1 in some data structure at index i
  for all $e_2 \in W_2$
    $i \leftarrow h_l((s_2^T - \hat{H}_2 e_2^T)/z)$ (ISD 2)
    $S \leftarrow$ read(i) //extract the elements stored at index i
    for all $e_1 \in S$
      if $\mathrm{wt}(s_2^T - \hat{H}_2(e_1 + e_2)^T) = t - p$ (ISD 3)
        return $(P, e_1 z y^{-1} + e_2)$. (SUCCESS)

Proposition 1: *If* $\binom{n}{t}(q-1)^t < q^r$ *(single solution), or if* $\binom{n}{t}(q-1)^t \geq q^r$ *(multiple solutions) and* $\binom{r}{t-p}\binom{k}{p}(q-1)^t \ll q^r$*, a lower bound for the expected cost (in number of operations) of the algorithm is*

$$WF_{qISD}(n, r, t, q) = \min_{l;p_1;p_2} \frac{N_{p;q}(l)}{\sqrt{q-1}} \cdot \left(\lambda_q^{-1} \left(\frac{2(q-1)l}{q\binom{l}{p_2'}(q-1)^{p_2'}} + p_2 \right) \times \sqrt{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}} + K_q \frac{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}{q^l} \right),$$

where

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}(q-1)^t}, \quad p = p_1 + p_2, \quad p_2' = \lfloor p_2/2 \rfloor, \quad \textit{and}$$

$$\lambda_q = 1 - \exp(-1) \approx 0.63.$$

An exception is $p = 0$, which corresponds to the classical ISD algorithm proposed by Prange (1962). In this case, we cannot gain a factor of $\sqrt{q-1}$, and the work factor is

$$WF_{qISD}(n, r, t, q) = \frac{\binom{n}{t}}{\binom{r}{t}}.$$

If $\binom{n}{t}(q-1)^t \geq q^r$ and $\binom{r}{t-p}\binom{k}{p}(q-1)^t \geq q^r$, the expected cost is

$$WF_{qISD}(n, r, t, q) \approx \min_{l;p} \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}(q-1)^{t-p+1}}}.$$

Proof: We only sketch the proof here and give full details in Appendix A.

By computing the number of syndromes that are tested in each iteration of the algorithm and the success probability of each error pattern we try, we can compute the expected number of iterations, which corresponds to the number of executions of the steps ISD 1 and ISD 2. While this number depends on the size of the sets W_1' and W_2, we can analytically find the optimal size for these sets. In addition to that, we compute the expected number of collisions we have to test per iteration (step ISD 3 in the algorithm). Multiplied by the number of iterations, this equals the number of executions of ISD 3. Finally, we use the number of operations required to execute ISD 1–3 to compute the total cost of the algorithm. □

The variable K_q represents the number of operations required to check condition ISD 3. A realistic value for K_q is $K_q = 2t$, which will be used for the parameters in Section 3.2. This value is justified by counting the computation of each row of $(s_2^T - \hat{H}_2(e_1 + e_2)^T)$ as one operation; since every row has the same probability of yielding 0 or 1, step ISD 3 ends after $2t$ rows on average.

In Finiasz and Sendrier (2009), the cost to compute the vectors in steps (ISD1) and (ISD2) was estimated to l operations. However, we can reduce the cost by optimising the calculation: by storing partial sums, we only have to add a single vector to compute the next syndrome. The number of operations required equals the weight of this vector, which is $l/2$ on average. In addition to that, we do a full, instead of a partial, Gaussian elimination, which corresponds to the central block I_l of H. These vectors contain only a single non-zero entry, which makes it particularly fast to add them in order to compute the syndrome: for every partial sum of p_1 columns, we get $\binom{l}{p_2}$ syndromes at a very low cost. These facts were described in Bernstein, Lange and Peters (2010).

The total work factor is the product of the number of iterations by the work factor per iteration. In practice, the latter is essentially the sum of a matrix multiplication (with the permutation matrix), the Gaussian elimination, and the search for collisions between L_1' and L_2'. Compared with the binary case, the Gaussian elimination is slower in the q-ary case, because every row has to be divided by the pivot entry. However, since matrix multiplication and Gaussian elimination are much faster than collision search, we allocate no cost to them.

3.2 Results

In Peters (2010), the author shows how to extend Lee-Brickell's and Stern's algorithms to code over $\mathbb{F}_q$. Website Peters (2010) lists the work factor of this algorithm against several parameters. We use the same parameters and compare these results with our lower bound.

Table 1 Table from Peters (2010), containing parameters for quasi-cyclic Berger et al. (2009) and quasi-dyadic Misoczki and Barreto (2009) codes

Code parameters				*Claimed*	$\log_2$(*# bit ops*)	*Lower bound*
q	*n*	*k*	*t*	*security level*	(*from Peters (2010)*)	$\log_2$(*#bit ops*)
256	459	255	50	80	81.93	65.05
256	510	306	50	90	89.43	72.94
256	612	408	50	100	102.41	86.50
256	765	510	50	120	101.58	85.14
1,024	450	225	56	80	83.89	62.01
1,024	558	279	63	90	91.10	69.17
1,024	744	372	54	110	81.01	57.88
4	2,560	1,536	128	128	181.86	178.78
16	1408	896	128	128	210.61	204.65
256	640	512	64	102	184.20	171.89
256	768	512	128	136	255.43	243.02
256	1024	512	256	168	331.25	318.65
2	2,304	1,281	64	80	83.38	80.60
2	3,584	1,536	128	112	112.17	109.98
2	4,096	2,048	128	128	136.47	134.53
2	7,168	3,073	256	192	215.91	214.72
2	8,192	4,096	256	256	265.01	264.13

Table 2 The two parameter sets from Bernstein, Lange and Peters (2010), which are below the lower bound proposed in Finiasz and Sendrier (2009) (number of bit operations in $\log_2$)

Code parameters				*Lower bound from Finiasz and Sendrier (2009)*	*# bit ops* (*from Bernstein, Lange) and Peters (2010)*	*Our new lower bound*
q	*n*	*k*	*t*			
2	6,624	5,129	117	255.18	254.15	251.95
2	30,332	22,968	494	999.45	996.22	991.94

For the algorithm from Peters (2010) as well as for our lower bound algorithm, the expected number of operations is the product of the number of iterations by the number of operations in each iteration. While the former factor is the same for both algorithms, or even a little higher for our algorithm, the lower bound for the number of operations per iteration is much smaller in our case, which results in the difference between these algorithms.

The following comparison is between our algorithm and the overlapping-sets version from Peters (2010), which is structurally closer to our algorithm than the even-split version. The runtime difference between these two versions is comparatively low.

3.2.1 Difference in the number of operations per iteration

The number of operations per iteration for the first algorithm is the sum of three steps:

1 reusing parts of information sets and performing precomputations

2 compute sums of p rows on l bits to calculate $\hat{H}_2 e^T$

3 for each collision (e_1, e_2), checking if $\mathrm{wt}(s_2^T - \hat{H}_2(e_1 + e_2)^T) = t - p$

To compare the cost of these steps with that used for our lower bound, we calculate all values for the $(450, 225, 56)$ parameter set over $\mathbb{F}_{1024}$. For this set, using $p = 1$, $l = 3$, $m = 1$, $c = 2$ and $r' = 1$ (the last variable is called r in Peters (2010), but it should not be confused with the co-dimension), we calculate a total cost of the first algorithm of $2^{76.9}$, which consists of $2^{56.1}$ iterations of $2^{20.8}$ operations.

Precomputations. The cost of the first step is given in Peters (2010) as

$$(n-1)\left((k-1)\left(1-\frac{1}{q^{r'}}\right)+(q^{r'}-r')\right)\frac{c}{r'},$$

where c and r' are algorithm parameters. For these parameters, this amounts to $2^{20.1}$ operations, so it is the most expensive step.

Our algorithm does not use precomputation, so we allocate no cost.

Computing sums of p rows to calculate $\hat{H}_2 e^T$. Cost of this step for the first algorithm:

$$((k-p+1)+(N+N')(q-1)^p)\,l.$$

The parameters N and N' are the sizes of the sets W_1 and W_2. For the parameters given above, this step adds $2^{19.4}$ operations. Our algorithm allocates to this step a cost of

$$\begin{aligned}&\frac{(q-1)l|W_1'|}{q\binom{l}{p_2'}(q-1)^{p_2'}}+|W_1'|p_2'+\frac{(q-1)l|W_2|}{q\binom{l}{p_2'}(q-1)^{p_2'}}+|W_2|p_2'\\ =&\left(\frac{2(q-1)l}{q\binom{l}{p_2'}(q-1)^{p_2'}}+p_2\right)\sqrt{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}.\end{aligned}$$

We make this optimistic (from the cryptanalyst's point of view) assumption for the cost of a matrix-vector multiplication to anticipate further software and hardware improvements for this operation. The result is $2^{4.9}$ operations in this case.

Checking collisions. The first algorithm allocates a cost of

$$\frac{q}{q-1}(t-2p)2p\left(1+\frac{q-2}{q-1}\right)\frac{NN'(q-1)^{2p}}{q^l}$$

to this step. For our set of parameters, this equals $2^{11.4}$ operations. In our algorithm, we expect the number of collisions to be

$$\frac{\lambda_q|W_1'|\cdot|W_2|}{q^l}=\frac{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}{q^l}.$$

The cost K_q of checking each collision is taken to be $K_q = 2t$. Since the expected number of collisions per iteration is very small, the expected cost per iteration is < 1.

Some of the assumptions above may seem fairly optimistic. However, we find it necessary to make these assumptions since we want to establish conservative lower bounds.

4 Impact of partial knowledge on ISD

We analyse two specific types of partial knowledge in this section, namely

1 error values come from a subset $E \subsetneq \mathbb{F}_q$

2 the entries of e are known, but their positions are not.

There are various situations when adversaries have partial knowledge of these types. For example, the NTRU cryptosystem can be attacked by an ISD-like algorithm; although the cryptosystem is defined over $\mathbb{F}_q$ (common values for q are 128 or 256), the random vector an attacker attempts to find is only ternary. Hence, an attacker knows that $E = \{1, -1\}$ (since the error values are the non-zero entries) and can exploit this knowledge.

An example for the second type of partial knowledge is any Stern-like identification scheme that works over $\mathbb{F}_q$, provided the scheme does not use measures to prevent information leakage. For one of the challenges sent by the verifier, the prover sends a random permutation of the secret key in the answer phase. This reveals the *value* of all entries of the secret, but not their positions.

There are other types of partial knowledge, e.g. knowledge about the structure of the underlying code, or that a solution is a regular word (a regular word consists of t blocks of size n/t, and each block contains exactly one non-zero entry). Also, other attacks (e.g. Generalised Birthday algorithms) should be able to exploit partial knowledge. We leave these research questions for future work.

4.1 Error values come from a set $E \subsetneq \mathbb{F}_q$

If we know that the error values come from a set $E \subsetneq \mathbb{F}_q$, we can limit the size of the sets W_i and L_i by redefining:

$$W_1 \subseteq E_0^{k+l} \cap \{e \in \mathcal{W}_{k+l;\lceil p/2\rceil;q} : \mathrm{le}(e) = 1\}, \tag{7}$$

$$W_2 \subseteq E_0^{k+l} \cap \{e \in \mathcal{W}_{k+l;\lfloor p/2\rfloor;q}\}, \tag{8}$$

$$L_1 = \left\{(\hat{H}_2 e^T)(\mathrm{le}(\hat{H}_2 e^T))^{-1} : e \in W_1\right\}, \tag{9}$$

$$L_2 = \left\{(s_2 - \hat{H}_2 e^T)(\mathrm{le}(s_2 - \hat{H}_2 e^T))^{-1} : e \in W_2\right\}, \tag{10}$$

where $E_0 = E \cup 0$. When restricting the leading entry of e to 1, the above assumes that $1 \in E$. If this is not the case, i.e. $1 \notin E$, any other element in E can be used to fix the leading entry of all vectors in W_1. This gives the following proposition:

Proposition 2 (Error values in a set E)**:** *If* $\binom{n}{t}|E|^t < q^r$ *(single solution), or if* $\binom{n}{t}|E|^t \geq q^r$ *(multiple solutions) and* $\binom{r}{t-p}\binom{k}{p}|E|^t \ll q^r$*, a lower bound for the expected cost (in number of operations) of the algorithm is*

$$WF_{qISD}(n,r,t,q) = \min_{l;p_1;p_2} N_{p;q}(l) \cdot \left(\lambda_q^{-1} \left(\frac{2|E|l}{(|E|+1)\binom{l}{p_2'}(q-1)^{p_2'}} + p_2 \right) \times \sqrt{\binom{k}{p_1}\binom{l}{p_2}|E|^{p-1}} + K_q \frac{\binom{k}{p_1}\binom{l}{p_2}|E|^{p-1}}{q^l} \right),$$

where

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}|E|^t, q^r\right)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}|E|^t}, \quad p = p_1 + p_2, \quad p_2' = \lfloor p_2/2 \rfloor, \quad \textit{and}$$
$$\lambda_q = 1 - \exp(-1) \approx 0.63.$$

The case $p = 0$ *is identical to the one in Proposition 1, hence*

$$WF_{qISD}(n,r,t,q) = \frac{\binom{n}{t}}{\binom{r}{t}}.$$

If $\binom{n}{t}|E|^t \geq q^r$ *and* $\binom{r}{t-p}\binom{k}{p}|E|^t \geq q^r$*, the expected cost is*

$$WF_{qISD}(n,r,t,q) \approx \min_{l;p} \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}|E|^{t-p+1}}}.$$

Proof: See Appendix B.

The difference to the proof of Proposition 1 is that we can decrease the size of the sets W_1' and W_2 by allowing only error patterns with values in E_0. The expected number of iterations is unchanged, but the number of operations per iteration decreases rapidly with E. □

4.2 Error values known (but not error positions)

Let the error values be $v_1, \ldots, v_t$. Depending on the size of the set $V := \{v_1, \ldots, v_t\}$, several strategies can be used. We will describe three cases:

1 $|V| = t$, i.e., all error values are different

2 $|V| < t$, but there are still many different error values

3 most error values belong to a small subset of V.

4.2.1 $|V| = t$

In this case, we can decrease the number of error vectors we have to try in each iteration in the following way. For each distribution of $p = p_1 + p_2$ errors in $k + l$ locations, we do not assign all combinations of q-ary values to the error positions, but instead randomly choose p out of the t known error values and assign them in all possible combinations. Hence, instead of

$$\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}$$

combinations, we only have to test

$$\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!.$$

For most parameters, the second expression is much smaller than the first. For these parameters, we redefine the sets W_i and L_i as follows

$$W_1 \subseteq \{e \in \mathcal{W}'_{k+l;\lceil p/2\rceil;q}\}, \tag{11}$$

$$W_2 \subseteq \{e \in \mathcal{W}'_{k+l;\lfloor p/2\rfloor;q}\}, \tag{12}$$

$$L_1 = \left\{(\hat{H}_2 e^T)(\mathrm{le}(\hat{H}_2 e^T))^{-1} : e \in W_1\right\}, \tag{13}$$

$$L_2 = \left\{(s_2 - \hat{H}_2 e^T)(\mathrm{le}(s_2 - \hat{H}_2 e^T))^{-1} : e \in W_2\right\}, \tag{14}$$

where $\mathcal{W}'_{k+l;p;q}$ is the set of all q-ary words of length $k + l$ and weight distribution $p_1 + p_2$, and those p values are taken from the known t error values. This gives the following result:

Proposition 3 (Known error values)**:** *If* $\binom{n}{t}t! < q^r$ *(single solution), or if* $\binom{n}{t}t! \geq q^r$ *(multiple solutions) and* $\binom{r}{t-p}\binom{k}{p}p!(t-p)! \ll q^r$*, a lower bound for the expected cost (in number of operations) of the algorithm is*

$$WF_{qISD}(n,r,t,q) = \min_{l;p_1;p_2} N_{p;q}(l) \cdot \left(\lambda_q^{-1}\left(\frac{2(q-1)l}{q\binom{l}{p'_2}\binom{t}{p'_2}p'_2!} + p_2\right)\right.$$
$$\left.\sqrt{\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!} + K_q\frac{\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!}{q^l}\right),$$

where

$$N_{p;q}(l) = \frac{\min(\binom{n}{t}t!, q^r)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!(t-p)!}, \quad p = p_1 + p_2, \quad p'_2 = \lfloor p_2/2\rfloor, \quad \textit{and}$$
$$\lambda_q = 1 - \exp(-1) \approx 0.63.$$

An exception is $p = 0$*, where the above method does not gain anything, hence*

$$WF_{qISD}(n,r,t,q) = \frac{\binom{n}{t}}{\binom{r}{t}}.$$

If $\binom{n}{t}t! \geq q^r$ *and* $\binom{r}{t-p}\binom{k}{p}p!(t-p)! \geq q^r$*, the expected cost is*

$$WF_{qISD}(n,r,t,q) \approx \min_{l;p} \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}(t-p)!}}.$$

Proof: See Appendix C.

Again, this type of partial knowledge allows to decrease the size of the sets W_1' and W_2. The method to construct these sets is different compared to the case above. Instead of restricting the allowed error values, we redefine the sets W_1' and W_2 such that the entries have the correct weight and weight distribution, and in addition, the entries are chosen from v_1 to v_t. This last condition is responsible for the additional binomial factors in the resulting formula of the lower bound. □

Remark 6: By the pigeonhole principle, the condition $|V| = t$ implies $t < q$ (since $v_i \neq 0$), so the above work factor is smaller than in the case without partial knowledge.

4.2.2 $|V| < t$, but still many different error values

This case enables further improvement compared to Section 4.2.1. Because of the condition $|V| < t$, some error values occur more than once. This further decreases the size of the $\mathcal{W}'$ sets, and hence W_i and L_i decrease in size.

For each of the $\binom{k+l}{p}$ patterns to distribute the errors, p values are chosen from V to be allocated to these error positions. If it happens that the same value is drawn more than once, then some permutations of these p error values lead to the same result, hence we can eliminate these.

The exact efficiency improvement depends not only on $|V|$ but on the detailed distribution of errors (e.g. three times the same error value is not the same as two pairs of the same value each). Let D_i, $i \in \mathbb{N}$, describe the distributions of error values that can occur after applying the random permutation, i.e. the number of pairs, triplets etc. Let $P(D_i)$ be the probability that such a distribution occurs, and $G(D_i)$ be the efficiency gain in such a case (with $0 < G(D_i) \leq 1$). Then the total work factor is decreased by a factor of

$$\sum_i P(D_i) \cdot G(D_i).$$

As an example, consider a case where $t = 20$, $p = 6$, $|V| = 10$, and $v_i = \lceil i/2 \rceil$; that is, all error values come in pairs. We can describe each distribution by the number j of error value pairs amongst the p last error positions, where collisions are searched for. Each pair decreases the number of operations by a factor of 2. The probability of the event is

$$P(j) = \frac{\binom{t/2}{j}\binom{t/2-j}{p-2j}2^{t/2-j}}{\binom{t}{p}}.$$

Hence, the number of operations performed by the algorithm is reduced to $\sum_{j=0}^{3} P(j) \cdot 2^{-j}$, an improvement of $\approx 36\%$.

In this example, we do not have a huge efficiency improvement, but for smaller V (and hence more pairs, triplets etc. of error values), the chance of a gain as well as the gain itself increases.

In order to analyse the situation in more detail, we introduce some notation: Let $v = |V|$ denotes the size of V; each of these v values occurs at $u_1, \ldots, u_v$ positions, and a configuration $(c_1, \ldots, c_v)$ with $\sum_i c_i = p$ denotes how many of these values are chosen for the last p error positions. Then the probability of a certain configuration $c = (c_1, \ldots, c_v)$ is

$$P(c) = \frac{\prod_i \binom{u_i}{c_i}}{\binom{t}{p}},$$

while the gain is

$$G(c) = \prod_j (c_i)!.$$

An interesting general case is $u_1 = \ldots = u_v = t/v$. Since p is usually small for ISD attacks, we do not expect a large efficiency gain if v is large as well. However, for small values of v, the gain increases significantly. See Table 3 for details.

Table 3 Expected efficiency gain for $t = 12$ and various values of p and v. Shown is the fraction of required operations between the improved and the standard algorithms

	$p = 2$	$p = 3$	$p = 4$
$v = 4$	0.91	0.74	0.53
$v = 3$	0.86	0.63	0.38
$v = 2$	0.72	0.44	0.12

Remark 7: Note that higher values of p allow for greater efficiency improvements. In fact, the example in Section 4.3 shows that the optimal value for p is significantly higher compared with the case without partial knowledge. This has to be taken into account when selecting attack parameters for the ISD algorithm.

4.2.3 Small V and most error values belonging to a small subset of V

If most error values come from a smaller subset $V_s \subset V$, we can modify the algorithm to take advantage of this fact. Instead of considering the full set V when defining sets L_i and searching for collisions, we can simply assume that the random permutation moved all errors with values not belonging to V_s to the left-hand side of the matrix. The sets L_i are then defined using the smaller set V_s, decreasing their size and thereby increasing the algorithm's efficiency. While this assumption decreases the probability of finding a suitable permutation, it is more than offset by the reduced number of operations per iteration.

4.3 Example 1: the CVE ID scheme

In Cayrel, Véron and El Yousfi Alaoui (2010), the authors propose a zero-knowledge identification scheme called CVE based on the q-ary syndrome decoding problem. In the

scheme, they propose to use a special permutation of the secret value s in order to hide the non-zero values of s. In this section, we recall the CVE scheme and study the effect of a leakage of the values of s on the ISD algorithm. In what follows, we write elements of $\mathbb{F}_q^n$ as n blocks of size $\lceil \log_2(q) \rceil = N$. We represent each element of $\mathbb{F}_q$ on N bits. We first introduce the special permutation that is used in the protocol to hide the non-zero values.

Definition 5: Let Σ be a permutation of $\{1, \ldots, n\}$ and $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{F}_q^n$ such that $\forall i, \gamma_i \neq 0$. We define the transformation $\Pi_{\gamma,\Sigma}$ as :

$$\begin{aligned} \Pi_{\gamma,\Sigma} : \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^n \\ v &\mapsto \left(\gamma_{\Sigma(1)} v_{\Sigma(1)}, \ldots, \gamma_{\Sigma(n)} v_{\Sigma(n)}\right) \end{aligned}$$

Notice that $\forall \alpha \in \mathbb{F}_q$, $\forall v \in \mathbb{F}_q^n$, it holds that $\Pi_{\gamma,\Sigma}(\alpha v) = \alpha \Pi_{\gamma,\Sigma}(v)$, and $\mathrm{wt}(\Pi_{\gamma,\Sigma}(v)) = \mathrm{wt}(v)$.

The CVE identification scheme consists of two parts: a key generation algorithm (Figure 1) and an identification protocol (Figure 2).

Figure 1 Key generation algorithm: parameters n, k, t, q are public

KeyGen:
Choose n, k, t, and q such that $\mathrm{WF}_{\mathrm{qISD}}(n, r, t, q) \geq 2^{\kappa}$
$H \xleftarrow{\$} \mathbb{F}_q^{r \times n}$
$e \xleftarrow{\$} \mathbb{F}_q^n$, s.t. $\mathrm{wt}(e) = t$.
$s \leftarrow He^T$
Output $(\mathrm{sk}, \mathrm{pk}) = (e, (s, H, t))$

Figure 2 Identification protocol

Prover $\mathcal{P}$		**Verifier** $\mathcal{V}$
	$(\mathrm{sk}, \mathrm{pk}) = (e, (s, H, t))$	
$u \xleftarrow{\$} \mathbb{F}_q^n, \Sigma \xleftarrow{\$} S_n$		
$\gamma \xleftarrow{\$} \mathbb{F}_q^{n*}$		
$c_1 \leftarrow h\left(\Sigma, \gamma, Hu^T\right)$		
$c_2 \leftarrow h\left(\Pi_{\gamma,\Sigma}(u), \Pi_{\gamma,\Sigma}(e)\right)$	$\xrightarrow{c_1, c_2}$	
	$\xleftarrow{\alpha}$	$\alpha \xleftarrow{\$} \mathbb{F}_q^*$
$\beta \longleftarrow \Pi_{\gamma,\Sigma}(u + \alpha e)$	$\xrightarrow{\beta}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, 1\}$
If $b = 0$:	$\xrightarrow{\Sigma, \gamma}$	**Check** $c_1 \stackrel{?}{=} h(\Sigma, \gamma, H\Pi_{\gamma,\Sigma}^{-1}(\beta)^T - \alpha s)$
Else:	$\xrightarrow{\Pi_{\gamma,\Sigma}(e)}$	**Check** $c_2 \stackrel{?}{=} h(\beta - \alpha\Pi_{\gamma,\Sigma}(e), \Pi_{\gamma,\Sigma}(e))$, $\mathrm{wt}(\Pi_{\gamma,\Sigma}(e)) \stackrel{?}{=} t$

Key generation. For $r := n - k$, the scheme uses a random $(r \times n)$ q-ary matrix H common to all users which corresponds to a parity check matrix of a random linear $[n, k]$ q-ary code. We can assume that H is described as $(I_r|R)$ where R is a random $r \times k$ matrix; as Gaussian elimination does not change the code generated by H, there is no loss of generality. Let κ be the security parameter. Figure 1 describes the key generation process.

Identification protocol. The secret key holder can prove his knowledge of e by using two blending factors: the transformation by means of a permutation and a random vector. The protocol has to be run several times to detect cheating provers. The security of the construction relies on the hardness of the general decoding problem, that is, on the difficulty of determining the preimage (with respect to left-hand multiplication by H) e of $s^T = He^T$. In Figure 2, h denotes a hash function and S_n the symmetric group of degree n.

4.3.1 Attack scenario

In this section, we show how an ISD attacker against the security of the CVE can gain partial knowledge on the secret. We analyse the CVE scheme and show that the prover leaks information to a potential attacker when using a 'classic' permutation instead of the function $\Pi_{\gamma,\Sigma}$.

We assume that attacker Eve attempts to impersonate the honest prover Alice. She can do this by recovering Alice's secret e; when Eve later runs the ID scheme with the honest verifier Bob, this allows her to successfully answer all queries.

First, Eve runs the ID scheme with Alice, where Eve sends the challenge $b = 1$. If the CVE scheme does not use the function $\Pi_{\gamma,\Sigma}$ to prevent information leakage, but rather sends the permuted secret, Eve will gain partial knowledge on the secret e.

Similarly to the original Stern scheme, Eve then attempts to find a code word w, minimising the weight of $(v + w)$ where $v \in H^{-1}(s)$ and s is Alice's public key. If she is successful, Eve can use $s' := v + w$ and randomly chosen u, γ and σ to successfully answer all queries by the verifier Bob. An element w with the above properties can be found using ISD-like algorithms.

Table 4 shows how much our modification of ISD attacks decreases the security of the CVE scheme (without the special permutation).

Table 4 Parameter sets for the CVE ID scheme (without the special permutation) over $\mathbb{F}_q$ for various field sizes

q	$[n, k, d]$	*Security level ISD only*	*Optimal value for p*	*Security level using improvements*	*Optimal value for p*
256	$[128, 64, 49]$	73.2	1	51.7	13
256	$[208, 104, 78]$	114.7	1	85.5	18

4.4 Example 2: NTRU scheme

The first version of the NTRUEncrypt Public Key Cryptosystem, which was simply called NTRU, was developed around 1996 by Hoffstein, Pipher and Silverman (1998).

NTRU is characterised by the three integers (N, p, q) such that:

- N is a prime number
- p a small integer
- q a large integer (often much larger than p), coprime with p
- d_f and d_g two positive integers.

In order to transfer the NTRU problem to a code-based problem over $\mathbb{F}_q$, we require that q is a prime power.

Bob's private key consists of two polynomials $f = \sum_{i=0}^{N-1} f_i X^i$ and $g = \sum_{i=0}^{N-1} g_i X^i$ in the ring $R \approx \mathbb{Z}[X]/(X^N - 1)$. The public key h is a polynomial of degree $N - 1$ such that $f * h = pg \bmod q$, where $*$ is the cyclic convolution.

Additionally, there are restrictions on f and g:

- f has d_f coefficients equal to 1, $d_f - 1$ coefficients equal to -1, and the remaining $N - (2d_f - 1)$ equal to 0.
- g has d_g coefficients equal to 1, d_g coefficients equal to -1, and the remaining $N - 2d_g$ equal to 0.

4.4.1 Attack of NTRU using ISD

We can translate the problem of finding the secret key h in the polynomial ring R to the problem of finding a vector in $\mathbb{F}_q$. The equation above thus becomes

$$H(G|F)^T = 0^T \quad \text{in } \mathbb{F}_q,$$

where $(G|F)$ represents the concatenation of $G = (g_0, \ldots, g_{N-1})$ and $F = (f_0, \ldots, f_{N-1})$, and H is the matrix of size $N \times 2N$:

$$H = \begin{pmatrix} -p & 0 & \ldots & 0 & h_0 & h_{N-1} & \ldots & h_1 \\ 0 & -p & \ldots & 0 & h_1 & h_0 & \ldots & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & -p & h_{N-1} & h_{N-2} & \ldots & h_0 \end{pmatrix}.$$

H is public while F and G are private. We know that F has $(2 \times d_f - 1)$ non-zero coefficients and G has $2 \times d_g$ non-zero coefficients. The non-zero coefficients of F and G constitute the private key.

This is the same situation as we have for many code-based schemes and we can directly apply ISD-based algorithms to find a set of $(2d_g + 2d_f - 1)$ columns of H whose weighted sum equals zero. In addition to that, however, we also know that the coefficients in this weighted sum are in $\{+1, -1\}$, so we do not have to search through the entire $\mathbb{F}_q$.

To our best knowledge, this is the first attack to use ISD and partial knowledge for cryptanalysing the NTRU scheme. The complexity of the attack here is larger than that of the best known attack against NTRU. However, we make the following point: partial knowledge does, in fact, reduce general running time for cryptanalytic algorithms, and it should thus be taken into consideration when designing cryptographic schemes.

5 Conclusion and outlook

Attacks based on ISD are amongst the most efficient generic attacks against code-based cryptosystems. In this paper, we have presented and proved lower bounds for ISD algorithms over $\mathbb{F}_q$. Part of the result is a modification of the algorithms from Finiasz and Sendrier (2009) which allows to increase the efficiency of the algorithm by a factor $\sqrt{q-1}$ compared to a straightforward q-ary generalisation of Finiasz and Sendrier (2009).

In addition, we described various situations where an attacker can obtain partial knowledge about the secret and we proposed two techniques to exploit this knowledge. These techniques were analysed and we proved new lower bounds for the complexity of the modified algorithms.

As an example, we show how to apply partial knowledge to attack the CVE identification scheme if information leakage is not prevented. The complexity of the attack is decreased significantly, even though we computed conservatively (e.g. we assumed all error values to be different). Clearly, the potential leakage of partial knowledge has to be analysed when designing new cryptosystems.

It is seen from Table 1 that the efficiency of concrete algorithms over $\mathbb{F}_2$ is not far from the lower bound, while over larger fields the gap is wider. We propose to further investigate improvements over $\mathbb{F}_q$ to decrease the size of this gap.

As future research, we suggest to analyse the impact of partial knowledge on other attacks, for example on Generalised Birthday algorithms. In addition to that, other types of partial knowledge should be considered. For example, in the FSB hash function Augot, Finiasz and Sendrier (2005), the 'message' is transformed into a regular word, which means that each block of size n/t has weight 1. It should be analysed how this knowledge can increase the efficiency of attacks in order to better estimate the security of cryptographic schemes.

Acknowledgements

We are most grateful to Richard Lindner, Cristina Onete and to our anonymous reviewers for their valuable comments during the preparation of this work and to Christiane Peters for our fruitful discussions.

References

Faugère, J.-C., Otmani, A., Perret, L. and Tillich, J.-P. (2010) 'Algebraic cryptanalysis of McEliece variants with compact keys – toward a complexity analysis', *SCC '10: Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography*, June 2010, RHUL, pp.45–55.

Andoni, A., Indyk, P., Nguyen, H.L. and Razenshteyn, I. (2014) 'Beyond locality-sensitive hashing', *SODA*, Portland, Oregon.

Augot, D., Finiasz, M. and Sendrier, N. (2005) 'A family of fast syndrome based cryptographic hash functions', in Dawson, E. and Vaudenay, S. (Eds.): *Mycrypt, Vol. 3715 of LNCS*, Springer, Germany, pp.64–83.

Barg, A. (1994) 'Some new NP-complete coding problems', *Problemy Peredachi Informatsii*, Vol. 30, pp.23–28 (in Russian).

Barg, A. (1997) 'Complexity issues in coding theory', *Electronic Colloquium on Computational Complexity (ECCC)*, Vol. 4, No. 46.

Becker, A., Joux, A., May, A. and Meurer, A. (2012) 'Decoding random binary linear codes in $2^{n/20}$: How 1 + 1 = 0 improves information set decoding', *Eurocrypt 2012*, Cambridge, UK.

Berger, T.P., Cayrel, P.-L., Gaborit, P. and Otmani, A. (2009) 'Reducing key length of the McEliece cryptosystem', in *AFRICACRYPT, Vol. 5580 of Lecture Notes in Computer Science*, Springer, Heidelberg, pp.77–97.

Berlekamp, E., McEliece, R. and van Tilborg, H. (1978) 'On the inherent intractability of certain coding problems', *IEEE Transactions on Information Theory*, Vol. 24, No. 3, pp.384–386.

Bernstein, D.J., Lange, T. and Peters, C. (2008) 'Attacking and defending the McEliece cryptosystem', *PQCrypto '08: Proceedings of the 2nd International Workshop on Post-Quantum Cryptography*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, pp.31–46.

Bernstein, D.J., Lange, T. and Peters, C. (2010) *Ball-Collision Decoding*, Cryptology ePrint Archive, Report 2010/585, http://eprint.iacr.org/2010/585.pdf.

Bernstein, D.J., Lange, T. and Peters, C. (2011) 'Smaller decoding exponents: ball-collision decoding', *Crypto 2011*, Santa Barbara, California, USA.

Canteaut, A. and Chabaud, F. (1998) '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*, Vol. 44, No. 1, pp.367–378.

Cayrel, P.-L., Véron, P. and El Yousfi Alaoui, S.M. (2010) 'Improved code-based identification scheme', in *SAC 2010*, Lecture Notes in Computer Science, Springer, Berlin.

Courtois, N., Finiasz, M. and Sendrier, N. (2001) 'How to achieve a McEliece-based digital signature scheme', in Boyd, C. (Ed.): *Asiacrypt 2001, Vol. 2248 in LNCS*, Springer-Verlag, Heidelberg, pp.157–174.

Dubiner, M. (2010) 'Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem', *IEEE Transactions on Information Theory*, Vol. 56, No. 8, pp.4166–4179.

Faugère, J.-C., Otmani, A., Perret, L. and Tillich, J.-P. (2010) 'Algebraic cryptanalysis of mceliece variants with compact keys', in Gilbert, H. (Ed.): *EUROCRYPT, Vol. 6110 of Lecture Notes in Computer Science*, Springer, Heidelberg, pp.279–298.

Finiasz, M. and Sendrier, N. (2009) 'Security bounds for the design of code-based cryptosystems', in Matsui, M. (Ed.): *Advances in Cryptology - ASIACRYPT 2009, Vol. 5912 in LNCS*, Springer, Heidelberg, pp.88–105.

Har-Peled, S., Indyk, P. and Motwani, R. (2012) 'Approximate nearest neighbor: towards removing the curse of dimensionality', *Theory of Computing*, Vol. 8, No. 1, pp.321–350.

Hoffstein, J., Pipher, J. and Silverman, J.H. (1998) 'NTRU: a ring-based public key cryptosystem', in Buhler, J. (Ed.): *ANTS, Vol. 1423 of Lecture Notes in Computer Science*, Springer, Heidelberg, pp.267–288.

Howgrave-Graham, N. and Joux, A. (2010) 'New generic algorithms for hard knapsacks', in *Advances in Cryptology-EUROCRYPT 2010, Vol. 6110 of Lecture Notes in Computer Science*, Springer, Berlin, pp.235–256.

Lee, P.J. and Brickell, E.F. (1988) 'An observation on the security of McEliece's public-key cryptosystem', *EUROCRYPT '88, Lecturer Notes in CS*, Springer, Berlin, Davos, Switzerland, pp.275–280.

Leon, J.S. (1988) 'A probabilistic algorithm for computing minimum weights of large error-correcting codes', *IEEE Transactions on Information Theory*, Vol. 34, No. 5, pp.1354–1359.

May, A., Meurer, A. and Thomae, E. (2011) 'Decoding random linear codes in o($2^{0.054n}$)', in *Advances in Cryptology - ASIACRYPT* 2011*, Lecture Notes in Computer Science*, Springer, Berlin.

May, A. and Ozerov, I. (2015) 'On computing nearest neighbors with applications to decoding of binary linear codes', in *Advances in Cryptology - EUROCRYPT* 2015*, Lecture Notes in Computer Science*, Springer, Berlin.

McEliece, R.J. (1978) *A Public-Key Cryptosystem Based on Algebraic Coding Theory*, DNS Progress Report, pp.114–116.

Minder, L. and Sinclair, A. (2009) 'The extended k-tree algorithm', in Mathieu, C. (Ed.): *SODA*, SIAM, Philadelphia, pp.586–595.

Misoczki, R. and Barreto, P.S.L.M. (2009) 'Compact McEliece keys from Goppa codes', *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Vol. 5867 of Lecture Notes in Computer Science*, Springer, Heidelberg.

Niebuhr, R., Cayrel, P.-L., Bulygin, S. and Buchmann, J. (2010a) 'Attacking code/lattice based cryptosystems using partial knowledge', in *Inscrypt 2010, Vol. 6584 of LNCS*, October 2010, Springer, Heidelberg.

Niebuhr, R., Cayrel, P.-L., Bulygin, S. and Buchmann, J. (2010b) 'On lower bounds for information set decoding over $\mathbb{F}_q$', in *SCC 2010*, RHUL, London, UK.

Persichetti, E. (2012) 'Compact McEliece keys based on quasi-dyadic srivastava codes', *Journal of Mathematical Cryptology*, Vol. 6, No. 2, pp.149–169.

Peters, C. (2010) 'Information-set decoding for linear codes over $\mathbb{F}_q$', in *PQCrypto, LNCS*, Springer, Heidelberg, pp.81–94.

Peters, C. (2010) *Iteration and Operation Count for Information-Set Decoding Over* $\mathbb{F}_q$, January 2010, http://www.win.tue.nl/ cpeters/isdfq.html.

Prange, E. (1962) 'The use of information sets in decoding cyclic codes', *IRE Transactions on Information Theory*, Vol. 8, No. 5, pp.5–9.

Shor, P.W. (1994) *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, IEEE Press, Washington, D.C. USA, pp.124–134.

Stern, J. (1989) 'A method for finding codewords of small weight', *Proceedings of Coding Theory and Applications*, Springer, London, pp.106–113.

Stern, J. (1993) 'A new identification scheme based on syndrome decoding', in Stinson, D.R. (Ed.): *CRYPTO, Vol. 773 of Lecture Notes in Computer Science*, Springer, New York, pp.13–21.

Umana, V.G. and Leander, G. (2009) *Practical Key Recovery Attacks on Two McEliece Variants*, Cryptology ePrint Archive, Report 2009/509. http://eprint.iacr.org/.

van Tilburg, J. (1988) 'On the McEliece public-key cryptosystem', in *CRYPTO '88, Lecturer Notes in CS*, Vol. 403, Springer, Berlin, pp.119–131.

Varshamov, R.R. (1957) 'Estimate of the number of signals in error correcting codes', *Doklady Akademii Nauk SSSR*, Vol. 117, pp.739–741. English translation in [21], pp.68–71.

Appendix A. Proof of Proposition 1

A.1 Efficiency improvement using the field structure of $\mathbb{F}_q$

The step of the algorithm that can be made more efficient using the field structure of $\mathbb{F}_q$ is the search for a pair (e_1, e_2) such that $e_1 \in \mathcal{W}_{k+l;\lfloor p/2 \rfloor;q}$, $e_2 \in \mathcal{W}_{k+l;\lceil p/2 \rceil;q}$ and

$$He_1^T = s_2^T - He_2^T,$$

where $\mathcal{W}_{k+l;p;q}$ is the set of all q-ary words of length $k+l$ and weight p.

Let W_1', W_2, L_1' and L_2' be defined as in (4)–(6). First note that for any pair (e_1, e_2) and all non-zero values $y \in \mathbb{F}_q$, we have

$$\hat{H}_2 e_1^T = s_2^T - \hat{H}_2 e_2^T \Leftrightarrow (\hat{H}_2 e_1^T) y^{-1} = (s_2^T - \hat{H}_2 e_2^T) y^{-1}.$$

Instead of storing $\hat{H}_2 e_1^T$ and $s_2^T - \hat{H}_2 e_2^T$, we can store $(\hat{H}_2 e_1^T)(\mathrm{le}(\hat{H}_2 e_1^T))^{-1}$ in L_1' and $(s_2^T - \hat{H}_2 e_2^T)(\mathrm{le}(s_2^T - \hat{H}_2 e_2^T))^{-1}$ in L_2', respectively. The list L_1', however, would contain every entry exactly $(q-1)$ times, since for every $y \in \mathbb{F}_q \backslash \{0\}$, e_1 and ye_1 yield the same entry. Therefore, we can generate the first list by using only vectors e_1 whose first non-zero entry is 1.

To see that there is exactly one collision between L_1' and L_2' for every solution of the problem, let (e_1, e_2) be a pair found by our algorithm. Let $y = \mathrm{le}(\hat{H}_2 e_1^T)$ and $z = \mathrm{le}(s^T - \hat{H}_2 e_2^T)$. Then we have

$$(\hat{H}_2 e_1^T) y^{-1} = (s^T - \hat{H}_2 e_2^T) z^{-1},$$

and therefore $(e_1 z y^{-1}, e_2)$ is a solution to the problem.

Conversely, let (e_1, e_2) be a solution to the problem, i.e., $\hat{H}_2 e_1^T + \hat{H}_2 e_2^T = s_2^T$. We want to show that there exists a collision between L_1' and L_2' which corresponds to this solution. Let $y = \mathrm{le}(\hat{H}_2 e_1^T)$ and $z = \mathrm{le}(s_2^T - \hat{H}_2 e_2^T)$. Since $\hat{H}_2 e_1^T = s_2^T - \hat{H}_2 e_2^T$, we have

$$(\hat{H}_2 e_1^T) y^{-1} = (s_2^T - \hat{H}_2 e_2^T) z^{-1}. \tag{15}$$

As we did not limit the set W_2, the right-hand side of Eq. (15) is an element of L_2'.

Let $x = \mathrm{le}(e_1)$. The first non-zero entry of $e_1' = e_1 x^{-1}$ is 1, so it was used to calculate one member of L_1'. As $\mathrm{le}(\hat{H}_2 e_1'^T) = \mathrm{le}(\hat{H}_2 (e_1 x^{-1})^T) = y x^{-1}$,

$$(\hat{H}_2 e_1'^T)(\mathrm{le}(\hat{H}_2 e_1'^T))^{-1} = (\hat{H}_2 (e_1 x^{-1})^T)(y x^{-1})^{-1} = (\hat{H}_2 e_1^T) y^{-1}.$$

Therefore, the left-hand side of Eq. (15) belongs to L_1'.

Since $z = y$, this collision between L_1' and L_2' corresponds to the solution (e_1, e_2).

Obviously, this improvement can only be applied if $p > 0$, i.e., if there actually is a search for collisions. If $p = 0$, we are simply trying to find a permutation which shifts all error positions into the first r positions of s, so the runtime is the inverse of the probability P_0 of this event with $P_0 = \binom{r}{t} / \binom{n}{t}$. For the rest of the appendix, we assume $p > 0$.

A.2 Cost of the algorithm

In most cases, the value of t will be smaller than the GV bound, and we expect the algorithm to require many iterations. In that case, in one iteration of our main loop, we expect to test a fraction

$$\begin{aligned}
\lambda_q(z_q) &= 1 - \left(1 - \frac{1}{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}\right)^{|W_1'| \cdot |W_2|} \\
&\approx 1 - \exp\left(-\frac{|W_1'| \cdot |W_2|}{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}\right) \\
&= 1 - \exp(-z_q)
\end{aligned}$$

of vectors in $\mathcal{W}_{k+l;p;q}$, where

$$z_q = \frac{|W_1'| \cdot |W_2|}{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}. \tag{16}$$

Even though the approximation through the exponential function slightly overestimates the success probability, the difference is less than 2^{-12} for relevant parameters.

The success probability of each pair (e_1, e_2) is the number of error combinations matching the syndrome in the last l rows, divided by the total number of possible values He^T with $e \in \mathcal{W}_{k+l;p;q}$. Depending on the code parameters, the latter is either given by the number of error patterns or by the number of syndromes:

$$P_q = \frac{\lambda_q(z_q)\binom{r-l}{t-p}(q-1)^{t-p+1}}{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}.$$

The success probability in one iteration of main loop is thus:

$$\begin{aligned} P_{p;q}(l) &= 1 - (1-P_q)^{|W_1'|\cdot|W_2|} \\ &\approx 1 - \exp(-P_q \cdot |W_1'| \cdot |W_2|) \\ &= 1 - \exp\left(-\frac{\lambda_q(z_q)}{N_{p;q}(l)}\right), \end{aligned}$$

where

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}{\binom{r-l}{t-p}|W_1'| \cdot |W_2|(q-1)^{t-p+1}}.$$

As described above, the set W_1' (and, similarly, W_2) is computed by storing partial sums, reducing the cost of each calculation to $l\frac{q-1}{q}$ operations. This is done for the rightmost $p_1' = \lceil p_1/2 \rceil$ error positions. Then the $p_2' = \lceil p_2/2 \rceil$ remaining positions are added, again storing partial sums, but at a cost of only one operation each, since the corresponding vectors have only one non-zero entry. If the sets are chosen to maximise this effect, every combination of p_1' error locations allows to compute $(\binom{l}{p_2'}(q-1)^{p_2'})$ vectors at the cost of only one operation. This leads to a total cost of

$$\frac{|W_1'|}{\binom{l}{p_2'}(q-1)^{p_2'}}\left(\frac{(q-1)l}{q} + \binom{l}{p_2'}(q-1)^{p_2'}p_2'\right) = \frac{(q-1)l|W_1'|}{q\binom{l}{p_2'}(q-1)^{p_2'}} + |W_1'|p_2'$$

to compute the syndromes corresponding to the vectors in W_1', and similarly for W_2.

For small parameters, the optimal p_2 might be odd, and the sets W_1' and W_2 will have different size. However, the difference between the optimal runtimes of an odd p_2 and $p_2 + 1$ is very small, thus we will assume an even p_2.

For small $P_{p;q}(l)$, the cost of the algorithm can be approximated to

$$\frac{N_{p;q}(l)}{\lambda_q(z_q)} \cdot \left(\frac{(q-1)l|W_1'|}{q\binom{l}{p_2'}(q-1)^{p_2'}} + |W_1'|p_2' + \frac{(q-1)l|W_2|}{q\binom{l}{p_2'}(q-1)^{p_2'}} + |W_2|p_2' + K_q\frac{\lambda_q(z_q)|W_1'| \cdot |W_2|}{q^l}\right),$$

which is the approximate number of iteration times the number of operations per iteration. K_q is the expected cost to perform and check that $\mathrm{wt}(s^T - H(e_1 + e_2)^T) = t - p$.

It is easy to see that choosing $|W_1'| = |W_2|$ minimises this expression.

$$N_{p;q}(l) \cdot \left(\frac{|W_1'|}{\lambda_q(z_q)} \left(\frac{2(q-1)l}{q\binom{l}{p_2'}(q-1)^{p_2'}} + p_2 \right) + K_q \frac{|W_1'|^2}{q^l} \right)$$
$$= \frac{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}{\binom{r-l}{t-p}|W_1'|^2(q-1)^{t-p}} \cdot \left(\frac{|W_1'|}{\lambda_q(z_q)} \left(\frac{2(q-1)l}{q\binom{l}{p_2'}(q-1)^{p_2'}} + p_2 \right) + K_q \frac{|W_1'|^2}{q^l} \right).$$

This expression decreases as the size of W_1' increases, so we maximise this size, choose $z_q = 1$ and set $\lambda_q = \lambda_q(1) = 1 - e^{-1}$. Using (16), we get

$$N_{p;q}(l) \cdot \left(\lambda_q^{-1} \left(\frac{2(q-1)l}{q\binom{l}{p_2'}(q-1)^{p_2'}} + p_2 \right) \right.$$
$$\left. \sqrt{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}} + K_q \frac{\binom{k}{p_1}\binom{l}{p_2}(q-1)^{p-1}}{q^l} \right),$$

where $N_{p;q}$ is now

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}(q-1)^t}.$$

Minimising over p_1, p_2, and l gives the result.

Now consider the case where $\binom{r}{t-p}\binom{k}{p}(q-1)^t \geq q^r$. Then the main loop is likely to succeed after a single iteration. This corresponds to the birthday algorithm described in Finiasz and Sendrier (2009):

$$\mathrm{WF}_{\mathrm{BA}} \approx \frac{2}{\sqrt{P}} \cdot \left(l + \frac{K_0}{2\sqrt{P}2^l} \right).$$

We can apply this result here, since it does not depend on the field size, but only on the success probability. In the q-ary case, this formula becomes

$$\mathrm{WF}_{\mathrm{qBA}} \approx \frac{2}{\sqrt{P}} \cdot \left(l + \frac{K_0}{2\sqrt{P}q^l} \right).$$

Easy analysis shows that the optimal value for l is

$$l = \log_q \left(\frac{\ln(q)K_0}{2\sqrt{P}} \right).$$

Applying this in our case with K_q instead of K_0 (since K_0 is the cost of the third step in the algorithm of Finiasz and Sendrier (2009), which corresponds to K_q when applied in the case of ISD), using

$$P = P_q \approx \frac{\binom{r-l}{t-p}(q-1)^{t-p+1}}{q^r},$$

and minimising over p and l yields the lower bound result:

$$\mathrm{WF}_{\mathrm{qISD}}(n,r,t,q) \approx \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}(q-1)^{t-p+1}}}.$$

Appendix B. Proof for Proposition 2

The proof follows the same approach as above, and we focus on the differences compared to that case.

Since the domain of the error values has been restricted from $\mathcal{W}_{n;p;q} \in \mathbb{F}_q^n$ to $\mathcal{W}_{n;p;q} \cap E^n$, every iteration of the algorithm tests a larger ratio of possible error vectors. We continue to denote this ratio by $\lambda_q(z_q) = 1 - \exp(z_q)$, where z_q changes and is now defined as

$$z_q = \frac{|W_1'| \cdot |W_2|}{\binom{k}{p_1}\binom{l}{p_2}|E|^{p-1}}. \tag{17}$$

This also changes the success probability P_q of each pair (e_1, e_2) to

$$P_q = \frac{\lambda_q(z_q)\binom{r-l}{t-p}|E|^{t-p+1}}{\min\left(\binom{n}{t}|E|^t, q^r\right)}.$$

Therefore, we get a success probability of each iteration of the main loop of

$$\begin{aligned} P_{p;q}(l) &= 1 - (1-P_q)^{\binom{k}{p_1}\binom{l}{p_2}|E|^p} \\ &\approx 1 - \exp\left(-P_q \cdot \binom{k}{p_1}\binom{l}{p_2}|E|^p\right) \\ &= 1 - \exp\left(-\frac{\lambda_q(z_q)}{N_{p;q}(l)}\right), \end{aligned}$$

with

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}|E|^t, q^r\right)}{\binom{r-l}{t-p}|W_1'| \cdot |W_2| \cdot |E|^t}.$$

Hence, for small $P_{p;q}(l)$, the cost of the algorithm is

$$\frac{\min\left(\binom{n}{t}(q-1)^t, q^r\right)}{\binom{r-l}{t-p}|W_1'|^2(q-1)^{t-p}} \cdot \left(\frac{|W_1'|}{\lambda_q(z_q)}\left(\frac{2|E|l}{(|E|+1)\binom{l}{p_2'}(q-1)^{p_2'}} + p_2\right) + K_q\frac{|W_1'|^2}{q^l}\right).$$

Using (16) and setting $z_q = 1$, we get

$$\begin{aligned} N_{p;q}(l) \cdot \Bigg(&\lambda_q^{-1}\left(\frac{2|E|l}{(|E|+1)\binom{l}{p_2'}(q-1)^{p_2'}} + p_2\right) \\ &\sqrt{\binom{k}{p_1}\binom{l}{p_2}|E|^{p-1}} + K_q\frac{\binom{k}{p_1}\binom{l}{p_2}|E|^{p-1}}{q^l}\Bigg), \end{aligned}$$

where $N_{p;q}$ is now

$$N_{p;q}(l) = \frac{\min\left(\binom{n}{t}|E|^t, q^r\right)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}|E|^t}.$$

Minimising over p_1, p_2 and l gives the result.

Similarly, in the case where $\binom{r}{t-p}\binom{k}{p}|E|^t \geq q^r$, we use the formula

$$\mathrm{WF}_{\mathrm{qBA}} \approx \frac{2}{\sqrt{P}} \cdot \left(l + \frac{K_0}{2\sqrt{P}q^l}\right)$$

with

$$P = P_q \approx \frac{\binom{r-l}{t-p}|E|^{t-p+1}}{q^r}.$$

Minimising over p and l yields the lower bound result:

$$\mathrm{WF}_{\mathrm{qISD}}(n,r,t,q) \approx \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}|E|^{t-p+1}}}.$$

Appendix C. Proof for Proposition 3

We mention only the changes compared with the proof in Appendix B.

Since the size of the sets W_i changes, the values of z_q, P_q and $N_{p;q}(l)$ become

$$z_q = \frac{|W_1| \cdot |W_2|}{\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!}, \tag{18}$$

$$P_q = \frac{\lambda_q\binom{r-l}{t-p}(t-p)!}{\min(\binom{n}{t}t!, q^r)},$$

$$N_{p;q}(l) = \frac{\min(\binom{n}{t}t!, q^r)}{\binom{r-l}{t-p}\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!(t-p)!}.$$

We can use the same strategy to efficiently compute the syndromes. In this case, the total cost is

$$\frac{|W_1'|}{\binom{l}{p_2'}\binom{t}{p_2'}p_2'!}\left(\frac{(q-1)l}{q} + \binom{l}{p_2'}\binom{t}{p_2'}p_2'! \cdot p_2'\right) = \frac{(q-1)l|W_1'|}{q\binom{l}{p_2'}\binom{t}{p_2'}p_2'!} + |W_1'|p_2'$$

to compute the syndromes corresponding to the vectors in W_1', and similarly for W_2.

Using (18) and following the same steps as above, we get

$$N_{p;q}(l) \cdot \left(\lambda_q^{-1} \left(\frac{2(q-1)l}{q\binom{l}{p_2'}\binom{t}{p_2'}p_2'!} + p_2 \right) \sqrt{\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!} + K_q \frac{\binom{k}{p_1}\binom{l}{p_2}\binom{t}{p}p!}{q^l} \right),$$

Minimising over l, p_1 and p_2 gives the result.

In the second case, we can use the same formula as above with the probability changed, i.e.,

$$\mathrm{WF}_{\mathrm{qBA}} \approx \frac{2}{\sqrt{P}} \cdot \left(l + \frac{K_0}{2\sqrt{P}q^l} \right).$$

With

$$l = \log_q \left(\frac{\ln(q)K_0}{2\sqrt{P}} \right),$$

$$P = P_q \approx \frac{\binom{r-l}{t-p}(t-p)!}{q^r}.$$

and minimising over p and l yields the lower bound result:

$$\mathrm{WF}_{\mathrm{qBA}}(n,r,t,q) \approx \frac{2lq^{r/2}}{\sqrt{\binom{r-l}{t-p}(t-p)!}}.$$

Minimising over p yields the result as above.

Notes

[1] Originally, the scheme is defined over $\mathbb{Z}_q$. Since it is irrelevant for our purpose whether we compute in a field or in a ring, we use the notation commonly used in code-based cryptography.

[2] SCC 2010 and Inscrypt 2010.