Information-set decoding for linear codes over $\mathbf{F}_q$

Christiane Peters

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
c.p.peters@tue.nl

Abstract. A code-based cryptosystem is considered secure if the best known attack against it is information-set decoding. Stern's algorithm and its improvements are well optimized and the complexity is reasonably well understood. However, these algorithms only handle codes over $\mathbf{F}_2$. This paper presents a generalization of Stern's information-set-decoding algorithm for decoding linear codes over arbitrary finite fields $\mathbf{F}_q$ and analyzes the complexity. This result makes it possible to compute the security of recently proposed code-based systems over non-binary fields. As an illustration, ranges of parameters for generalized McEliece cryptosystems using classical Goppa codes over $\mathbf{F}_{31}$ are suggested for which the new information-set-decoding algorithm needs 2^{128} bit operations.
Keywords: Generalized McEliece cryptosystem, security analysis, Stern attack, linear codes over $\mathbf{F}_q$, information-set decoding.

1 Introduction

Quantum computers will break the most popular public-key cryptosystems. The McEliece cryptosystem — introduced by McEliece in 1978 [10] — is one of the public-key systems without known vulnerabilities to attacks by quantum computers. Grover's algorithm can be countered by doubling the key size (see [6], [12]). Its public key is a random-looking algebraic code over a finite field. Encryption in McEliece's system is remarkably fast. The sender simply encodes a plaintext and adds some errors. The receiver, having generated the code by secretly transforming a Goppa code, can use standard Goppa-code decoders to correct the errors and recover the plaintext.

The security of the McEliece cryptosystem relies on the fact that the published code does not come with any known structure. An attacker is faced with the *classical decoding problem*: Find the closest codeword in a linear code C to a given vector in the ambient space of C, assuming that there is a unique closest codeword. This is a well known-problem. Berlekamp, McEliece, and van Tilborg [1] showed that the general decoding problem for linear binary codes is NP-complete. Moreover, the classical decoding problem is assumed to be hard on average, i.e., there are no weak instances.

Information-set decoding. An attacker does not know the secret code and thus has to decode a random-looking code without any obvious structure. The best known algorithms which do not exploit any code structure rely on information-set decoding, an approach introduced by Prange in [13]. The idea is to find a set of coordinates of a garbled vector which are error-free and which correspond to an invertible submatrix of the code's generator matrix. Then, the original message can be computed by multiplying the encrypted vector by the inverse of the submatrix. Improvements of this simplest form of information-set decoding were devised by Lee and Brickell [8], Leon [9], and Stern [14] — all for binary linear codes.

* Date of this document: 2009.12.01. This work has been supported in part by the European Commission through the ICT Programme under Contract ICT–2007–216676 ECRYPT II.

Best known attacks against binary McEliece. At PQCrypto 2009 Bernstein, Lange and Peters [2] presented several improvements to Stern's attack and gave a precise analysis of the complexity. Finiasz and Sendrier [5] presented a further improvement which can be combined with the improvements in [2] but did not analyze the combined attack. [2] suggests the use of binary Goppa codes of length 2960 and dimension 2288 with a degree-56 Goppa polynomial and 57 added errors for 128-bit security.

Decreasing public-key sizes by using larger fields. Recently, base fields other than $\mathbf{F}_2$ were suggested, e.g., [11]. This idea is interesting as it has the potential to reduce the public-key size. One could hope that using a code over $\mathbf{F}_q$ saves a factor of $\log_2 q$: row and column dimension of the generator matrix both shrink by a factor of $\log_2 q$ at the cost of the matrix entries having size $\log_2 q$. However, information-set-decoding algorithms do not scale as brute force attacks. It is important to understand the implications of changing from $\mathbf{F}_2$ to $\mathbf{F}_q$ for arbitrary prime powers q on the attacks. Note that some papers claim structural attacks against [11] but they are using that the codes are dyadic and do not attack the general principle of using larger base fields.

Contributions of this paper. This paper generalizes Lee–Brickell's algorithm and Stern's algorithm to decoding algorithms for codes over arbitrary fields and extends the improvements from [2] and [5]. The most important contribution is a precise analysis of these improved and generalized algorithms. For $q = 31$, code parameters (length n, dimension k and error-correction capacity t of the non-subfield Goppa code) are presented that require 2^{128} bit operations to compute the closest codeword.

Acknowledgments. The author would like to thank Dan Bernstein and Tanja Lange for fruitful discussions and in particular Tanja for helpful suggestions and detailed comments in the writing of this paper.

2 The McEliece cryptosystem

This section gives the background on the McEliece cryptosystem and introduces notation for linear codes which is used throughout this paper.

Linear codes. Let $\mathbf{F}_q$ be a finite field with q elements. An $[n, k]$ *code* over $\mathbf{F}_q$ is a linear code of length n and dimension k, i.e., a k-dimensional subspace of $\mathbf{F}_q^n$.

An $[n, k]$ code C is given by a *generator matrix* which is a $k \times n$ matrix G such that $C = \{\mathbf{m}G : \mathbf{m} \in \mathbf{F}_q^k\}$. The *parity-check matrix* of an $[n, k]$ code C is an $(n-k) \times n$ matrix H such that $C = \{\mathbf{c} \in \mathbf{F}_q^n : H\,\mathbf{c}^T = 0\}$.

The matrix G corresponds to a map $\mathbf{F}_q^k \to \mathbf{F}_q^n$ sending a message $\mathbf{m}$ of length k to a vector in $\mathbf{F}_q^n$. By definition $GH^T = 0$. Given the generator matrix G of a linear code C one can easily determine a matching parity check matrix H by linear transformations. In particular, if G is given in *systematic form*, i.e., $G = (I_k|Q)$ where Q is a $k \times (n-k)$ matrix then $H = (-Q^T|I_{n-k})$ is a parity check matrix for the code $\mathbf{F}_q^k G$.

The *Hamming distance* between two words in $\mathbf{F}_q^n$ is the number of coordinates where they differ. The *Hamming weight* of a word is the number of non-zero coordinates. The *minimum distance* of a linear code C is the smallest Hamming weight of a nonzero codeword in C.

Classical Goppa codes. Fix a finite field $\mathbf{F}_q$. Fix a positive integer $m > 1$ and consider $\mathbf{F}_{q^m}$. Fix elements $a_1, \ldots, a_n$ in $\mathbf{F}_{q^m}$ and a polynomial $g(z)$ in $\mathbf{F}_{q^m}[z]$ of degree t such that $mt < n$ and such that $g(a_i) \neq 0$ for all i. The polynomial $g(z)$ is called *Goppa polynomial*.

The words $c = (c_1, \ldots, c_n)$ in $\mathbf{F}_{q^m}^n$ with

$$\sum_{i=1}^{n} \frac{c_i}{z - a_i} \equiv 0 \mod g(z) \tag{1}$$

form an $[n, n-t]$ code C in $\mathbf{F}_{q^m}^n$. The Goppa code $\Gamma(a_1, \ldots, a_n, g)$ is the restriction of C to the field $\mathbf{F}_q$, i.e., the set of elements $(c_1, \ldots, c_n)$ in $\mathbf{F}_q^n$ which satisfy (1). As a subfield subcode of C the code $\Gamma(a_1, \ldots, a_n, g)$ has dimension $k \geq n - mt$. Its minimum distance is at least $t + 1$ for general fields $\mathbf{F}_q$ and at least $2t + 1$ if $q = 2$. See [7] for larger lower bounds for non-binary fields.

Let $\Gamma(a_1, \ldots, a_n, g)$ be a Goppa code of length n where $a_1, \ldots, a_n$ are elements in a degree-m extension $\mathbf{F}_{q^m}$ of $\mathbf{F}_q$ and g is a Goppa polynomial of degree t. Assume that $\Gamma(a_1, \ldots, a_n, g)$ has dimension exactly $n - mt$. Fix a basis of $\mathbf{F}_{q^m}$ over $\mathbf{F}_q$ and write each element of $\mathbf{F}_{q^m}$ with respect to that basis. Then, a parity check matrix for $\Gamma(a_1, \ldots, a_n, g)$ is given by the following $mt \times n$ matrix over $\mathbf{F}_q$

$$H = \begin{pmatrix} 1/g(\alpha_1) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{pmatrix},$$

where each entry is a vector written in the chosen $\mathbf{F}_q$-basis of $\mathbf{F}_{q^m}$.

The code $\Gamma(a_1, \ldots, a_n, g)$ is often referred to as a "classical" Goppa code since it is the basic construction of a genus-0 geometric Goppa code which Goppa later generalized for higher-genus varieties.

Set-up of the McEliece cryptosystem. The *secret key* of the McEliece cryptosystem consists of a classical Goppa code $\Gamma = \Gamma(a_1, \ldots, a_n, g)$ over a finite field of $\mathbf{F}_q$ of length n and dimension k with an error-correction capacity of w errors. A generator matrix G for the code Γ as well as an $n \times n$ permutation matrix P, and an invertible $k \times k$ matrix S are randomly generated and kept secret as part of the secret key.

The parameters n, k, and w are *public system parameters*. The *McEliece public key* is the $k \times n$ matrix $\hat{G} = SGP$.

McEliece encryption of a message $\mathbf{m} \in \mathbf{F}_q^k$: Compute $\mathbf{m}\hat{G}$. Then hide the message by adding a random error vector $\mathbf{e}$ of length n and weight w. Send $\mathbf{y} = \mathbf{m}\hat{G} + \mathbf{e}$.

McEliece decryption: Compute $\mathbf{y}P^{-1} = \mathbf{m}SG + \mathbf{e}P^{-1}$. Use the decoding algorithm to find $\mathbf{m}S$ and thereby $\mathbf{m}$.

The decryption algorithm works since $\mathbf{m}SG$ is a codeword in Γ and the vector $\mathbf{e}P^{-1}$ has weight w.

An attacker who got hold of an encrypted message $\mathbf{y}$ has two possibilities in order to retrieve the original message $\mathbf{m}$.

- Find out the secret code; i.e., find G given $\hat{G}$.
- Or decode $\mathbf{y}$ without knowing an efficient decoding algorithm for the public code given by $\hat{G}$.

Attacks of the first type are called *structural attacks*. If G or an equivalently efficiently decodable representation of the underlying code can be retrieved in subexponential time, this

code should not be used in the McEliece cryptosystem. Suitable codes are such that the best known attacks are decoding random codes. In the next section we will describe how to correct errors in a random-looking code with no obvious structure.

3 Generalizations of information-set-decoding algorithms

This section generalizes two information-set-decoding algorithms. Lee–Brickell's algorithm and Stern's algorithm — both originally designed for binary codes — are stated for arbitrary finite fields $\mathbf{F}_q$. Stern's algorithm is more efficient and supersedes Lee–Brickell's algorithm but the latter is easier to understand and the generalization of it can be used as a stepping stone to the generalization of Stern's.

The preliminaries are the following: Let C be an $[n, k]$ code over a finite field $\mathbf{F}_q$. Let G be a generator matrix for C. Let I be a non-empty subset of $\{1, \ldots, n\}$. Denote by G_I the restriction of G to the columns indexed by I. For any vector $\mathbf{y}$ in $\mathbf{F}_q^n$ denote by $\mathbf{y}_I$ the restriction of $\mathbf{y}$ to the coordinates indexed by I.

Let $\mathbf{m}$ be a vector in $\mathbf{F}_q^k$ and $\mathbf{c} = \mathbf{m}G$. Let $\mathbf{y}$ be a vector in $\mathbf{F}_q^n$ at distance w from $\mathbf{c}$. The goal of this section is to determine a vector $\mathbf{e} \in \mathbf{y} + \mathbf{F}_q^k G$ of weight w given G, $\mathbf{y}$ and w.

We start with the definition of an information set.

Information-set decoding. Let G^{sys} be a generator matrix for C in systematic form and $\mathbf{c} = \mathbf{m}G^{sys}$ for some vector $\mathbf{m}$ in $\mathbf{F}_q^k$. Since the first k columns of G^{sys} form the identity matrix the first k positions of $\mathbf{c}$ equal $\mathbf{m}$. The first k symbols of $\mathbf{m}G^{sys}$ are therefore called *information symbols*.

The notion of information symbols leads to the concept of information sets as follows. Let G be an arbitrary generator matrix of C. Let I be a size-k subset of $\{1, \ldots, n\}$. The columns indexed by I form a $k \times k$ submatrix of G which is denoted by G_I. If G_I is invertible the I-indexed entries of any codeword $\mathbf{m}G_I^{-1}G$ are information symbols and the set I is called an *information set*. Note that $G_I^{-1}G$ and G generate the same code.

Information-set decoding in its simplest form takes as input a vector $\mathbf{y}$ in $\mathbf{F}_q^n$ which is known to have distance w from a codeword $\mathbf{c}$ in C. Let I be an information set. Assume that $\mathbf{y}$ and $\mathbf{c}$ coincide on the positions indexed by I, i.e., no errors occurred at these positions. Then, $\mathbf{y}_I G_I^{-1}$ is the preimage of $\mathbf{c}$ under the linear map induced by G and we obtain $\mathbf{c}$ as $(\mathbf{y}_I G_I^{-1})G$.

The following subsection presents Lee–Brickell's algorithm which is a classical information-set-decoding algorithm and serves as a basis for all further improvements.

Notation: for any a in an information set I let $\mathbf{g}_a$ denote the unique row of $G_I^{-1}G$ where column a has a 1.

Lee–Brickell's algorithm. Let p be an integer with $0 \leq p \leq w$.

1. Choose an information set I.
2. Replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_I G_I^{-1} G$.
3. For each size-p subset $A = (a_1, \ldots, a_p) \subset I$: For each $\mathbf{m} = (m_1, \ldots, m_p)$ in $\mathbf{F}_q^p$: Compute $\mathbf{e} = \mathbf{y} - \sum_{i=1}^{p} m_i \mathbf{g}_{a_i}$. If $\mathbf{e}$ has weight w print $\mathbf{e}$. Else go back to Step 1.

Step 1 can be performed by choosing k indices in $\{1, \ldots, n\}$ uniformly at random and then performing Gaussian elimination on G in order to see if its I-indexed columns form an invertible submatrix G_I. A better way of determining an information set I is to choose k

columns one by one: check for each newly selected column if it does not linearly depend on the already selected columns.

If $p = 0$ Step 3 consists only of checking whether $\mathbf{y} - \mathbf{y}_I G_I^{-1} G$ has weight w. If $p > 0$ Step 3 requires going through all possible weighted sums of p rows of G which need to be subtracted from $\mathbf{y} - \mathbf{y}_I G_I^{-1} G$ in order to make up for the p errors permitted in I.

Steps 1–3 form one iteration of the generalized Lee–Brickell algorithm. If the set I chosen in Step 1 does not lead to a weight-w word in Step 3 another iteration has to be performed.

The parameter p is chosen to be a small number to keep the number of size-p subsets small in Step 3. In the binary case $p = 2$ is optimal; see e.g., [3].

Lee–Brickell, Leon, Stern. Stern's algorithm was originally stated to be a minimum-weight-word-finding algorithm for binary linear codes. Following [3] Stern's algorithm is stated as a fixed-distance-decoding algorithm. Given $\mathbf{y}$ in $\mathbf{F}_q^n$, G and w: find $\mathbf{e}$ such that $\mathbf{e}$ lies in $\mathbf{y} + \mathbf{F}_q^k G$. This algorithm is still a minimum-weight-word-finding algorithm: just choose $\mathbf{y}$ to be the zero-codeword in $\mathbf{F}_q^n$ and the algorithm yields a desired codeword of given length w. The basic Stern algorithm uses two parameters p and ℓ whose size are determined later on.

In each round an information set I is chosen. Stern's algorithm uses the idea of Lee and Brickell to allow a fixed number of errors in the information set. The algorithm also uses the idea of Leon's minimum-weight-word-finding algorithm [9] to look for the error vector $\mathbf{e}$: since $\mathbf{e}$ is a low-weight vector one restricts the number of possible candidates to those vectors having ℓ zeros outside the I-indexed columns.

Stern's algorithm divides the information set into two equal-size subsets X and Y and looks for words having exactly weight p among the columns indexed by X, exactly weight p among the columns indexed by Y and exactly weight 0 on ℓ positions outside the I-indexed columns.

Stern's algorithm. Let p be an integer with $0 \leq p \leq w$. Let ℓ be an integer with $0 \leq \ell \leq n-k$. For simplicity assume that k is even.

1. Choose an information set I.
2. Replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_I G_I^{-1} G$.
3. Choose a uniform random subset $X \subset I$ of size $k/2$.
4. Set $Y = I \setminus X$.
5. Select a uniform random size-ℓ subset Z in $\{1, \ldots, n\} \setminus I$.
6. For any size-p subset $A = \{a_1, \ldots, a_p\} \subset X$: Consider the set $\mathcal{V}_A = \left\{\mathbf{y} - \sum_{i=1}^{p} m_i \mathbf{g}_{a_i} : \mathbf{m} = (m_1, \ldots, m_p) \in (\mathbf{F}_q^*)^p\right\}$. For each $\phi \in \mathcal{V}_A$ compute the vector $\phi(Z) \in \mathbf{F}_q^\ell$: the Z-indexed entries of ϕ.
7. For any size-p subset $B = \{b_1, \ldots, b_p\} \subset Y$: Consider the set $\mathcal{V}_B = \left\{\sum_{j=1}^{p} m'_j \mathbf{g}_{b_j} : \mathbf{m}' = (m'_1, \ldots, m'_p) \in (\mathbf{F}_q^*)^p\right\}$. For each $\psi \in \mathcal{V}_B$ compute the vector $\psi(Z) \in \mathbf{F}_q^\ell$: the Z-indexed entries of ψ.
8. For each pair (A, B) where there is a pair of vectors ϕ and ψ such that $\phi(Z) = \psi(Z)$: Compute $\mathbf{e} = \mathbf{y} - \sum_i m_i \mathbf{g}_{a_i} - \sum_j m'_j \mathbf{g}_{b_j}$. If $\mathbf{e}$ has weight w print $\mathbf{e}$. Else go back to Step 1.

This algorithm finds a weight-w vector $\mathbf{e}$ in $\mathbf{y} + \mathbf{F}_q^k G$ if an information set I together with sets X, Y, and Z can be found such that $\mathbf{e}$ has weight p, p, 0 on the positions indexed by X, Y, and Z. Steps 1–8 form one iteration of the generalized Stern algorithm. If the set I chosen in Step 1 does not lead to a weight-w word in Step 8 another iteration has to be performed.

4 Analysis of an improved version of Stern's algorithm for prime fields

This section analyzes the cost for the generalization of Stern's algorithm as presented in Section 3. In this section the field $\mathbf{F}_q$ is restricted to prime fields. The general case is handled in the next section.

Note that the Stern algorithm stated in Section 3 is the basic algorithm. The following analysis takes several speedups into account that were introduced in [2] for the binary case.

Success probability of the first iteration. Note that in the basic Stern algorithm the information set I is chosen uniformly at random. I.e., k columns are chosen and then Gaussian elimination is performed. Stern proposed to choose the columns one by one and to perform Gaussian elimination after each selection in order to check whether the linearly independence is still given after taking a new column.

The success probability of a randomly chosen information set I to have p errors among the columns indexed by a size-$(k/2)$ subset $X \subset I$, p errors among the size-$(k/2)$ subset $Y \subset I$ and ℓ zeros among the columns indexed by the size-ℓ subset Z is given by $\binom{k/2}{p}^2 \binom{n-k-\ell}{w-2p} \Big/ \binom{n}{w}$; see e.g., [3].

Reusing parts of information sets and precomputations. Canteaut and Chabaud in [4] proposed to use a column-swapping technique for Stern's algorithm in order to cut down on Gaussian elimination costs. If an information set I does not lead to a weight-w vector $\mathbf{e}$ then instead of abandoning the whole set I reuse $k-1$ columns and select a new column out of the remaining $n-k$ non-selected columns of G. The submatrix $G_{I'}$ has to be updated for this new information set I'. Bernstein, Lange, and Peters pointed out in [2] that selecting only a single new column increases the number of iterations of Stern's algorithm significantly and proposed to swap more than one column in each round: Reuse $k-c$ columns from the previous iteration and select c new linearly independent columns out of the non-selected $n-k$ columns. The value c has to be chosen with respect to the code length n, its dimension k and the error weight w.

At the beginning of each new iteration there are k columns from the previous iteration in row echelon form. Exchanging c columns means that Gaussian elimination has to be performed on those c columns.

Updating the matrix and looking for pivots in c columns with respect to the chosen columns is done using precomputations. Since sums of certain rows are used multiple times those sums are precomputed. Following [2, Section 4] pick e.g., the first r rows and compute all possible sums of those rows. The parameter r needs to be tuned with respect to the field size q and the code dimension k. In particular, $r \leq c$. Starting with r columns one has to precompute only $q^r - r - 1$ sums of r rows. Each of the remaining $k-r$ rows requires on average $1 - 1/q^r$ vector additions.

Compute sums of p rows. In Step 6 one has to compute $\binom{k/2}{p}(q-1)^p$ vectors $\mathbf{y} - \sum_{i=1}^{p} m_i \mathbf{g}_{a_i}$ on ℓ positions. Computing those vectors naively one by one would require $p\ell$ multiplications and $p\ell$ additions in $\mathbf{F}_q$ per vector. A better way to do this is to first compute $\frac{k}{2} - p + 1$ vectors $\mathbf{y} - \mathbf{g}_i$ with $i \in X$ and then to build up all needed sums of p rows by using intermediate sums. This saves a factor of p. Computing all possible vectors induced by subsets A on the Z-indexed columns boils down to one row addition for each vector, i.e., ℓ additions in $\mathbf{F}_q$ per vector. In Step 7 the same trick is applied to compute all vectors in induced by size-p subsets of Y on the Z-indexed columns.

Collisions. The expected number of colliding vectors $\phi(Z)$, $\psi(Z)$ is about $\left(\binom{k/2}{p}(q-1)^p\right)^2 \Big/ q^\ell$. For each collision one computes $\mathbf{y}$ minus the sum of $2p$ weighted rows on all positions outside X, Y, and Z. Naive computation of one such a vector would take $2p$ multiplications and $2p$ additions on $n-k-\ell$ positions. First of all one can discard multiplications by 1, leaving $2p$ additions and $(2p)(q-2)/(q-1)$ multiplications. Looking more carefully one observes that each entry has a chance of $(q-1)/q$ to be a non-zero entry. In order to save operations, one computes the result in a column-by-column fashion and uses an early abort: After about $(q/(q-1))(w-2p)$ columns were handled it is very likely that the resulting row vector has more than the allowed $w-2p$ non-zero entries and can be discarded. This means that partial collisions that do not lead to full collision consume only $(q/(q-1))(w-2p)$ operations.

To estimate the costs per iteration we need to express the costs in one measure, namely in additions in $\mathbf{F}_q$. We described Steps 6–8 using multiplications. Since we consider only quite small fields $\mathbf{F}_q$ multiplications can be implemented as table lookups and thus cost the same as one addition

Cost for one iteration of Stern's algorithm. Stern's algorithm in the version presented here uses parameters p, ℓ and additional parameters c and r.

The cost of one iteration of Stern's algorithm is as follows:

$$\begin{aligned}
&(n-1)\left((k-1)\left(1-\frac{1}{q^r}\right)+(q^r-r)\right)\frac{c}{r}\\
&+\left(\left(\frac{k}{2}-p+1\right)+2\binom{k/2}{p}(q-1)^p\right)\ell\\
&+\frac{q}{q-1}(w-2p)2p\left(1+\frac{q-2}{q-1}\right)\frac{\binom{k/2}{p}^2(q-1)^{2p}}{q^\ell}.
\end{aligned}$$

Choice of parameters for Stern's algorithm. The parameter p is chosen quite small in order to minimize the cost of going though all subsets A, B of X and Y. The parameter ℓ is classically chosen to balance the number of all possible length-ℓ vectors $\phi(Z)$ and $\psi(Z)$, $2\binom{k/2}{p}(q-1)^p$ with the number of expected collisions on ℓ positions, $\binom{k/2}{p}^2(q-1)^{2p}\Big/ q^\ell$. A reasonable choice is

$$\ell = \log_q \binom{k/2}{p} + p\log_q(q-1).$$

5 Analysis of an improved version of Stern's algorithm for extension fields

We presented a generalization for information-set decoding over arbitrary finite fields $\mathbf{F}_q$. However, the cost analysis in the previous section was restricted to prime values of q. Here we point out the differences in handling arbitrary finite fields.

The main difference in handling arbitrary finite fields is in Steps 6 and 7 of the generalized Stern algorithm when computing sums of p rows coming from subsets A of X, and sums of p rows coming from subsets B of Y. In prime fields all elements are reached by repeated addition since 1 generates the additive group. If q is a prime power 1 does not generate the additive group.

Let $\mathbf{F}_q$ be represented over its prime field via an irreducible polynomial $h(x)$. To reach all elements we also need to compute x times a field element, which is essentially the cost

of reducing modulo h. In turn this means several additions of the prime field elements. Even though these operations technically are not additions in $\mathbf{F}_q$, the costs are essentially the same. This means that the costs of these steps are the same as before.

In the analysis of Step 8 we need to account for multiplications with the coefficient vectors $(m_1, \ldots, m_p)$ and $(m'_1, \ldots, m'_p)$. This is the same problem that we faced in the previous section and thus we use the same assumption, namely that one multiplication in $\mathbf{F}_q$ has about the same cost as one addition in $\mathbf{F}_q$.

This means that a good choice of ℓ again is given by

$$\ell = \log_q \binom{k/2}{p} + p \log_q(q-1).$$

6 Increasing the collision probability in Stern's algorithm

In [5] Finiasz and Sendrier proposed a speedup of Stern's algorithm. This section generalizes this approach to codes over arbitrary finite fields.

Stern splits an information set I into two disjoint sets X and Y, each of size $\binom{k/2}{p}$ and searches for collisions among size-p subsets taken from X and Y.

Finiasz and Sendrier propose not to split the information set I into two disjoint sets but to look more generally for collisions. The split of I into two disjoint size-$(k/2)$ sets is omitted at the benefit of creating more possible words having weight $2p$ among the information set.

This version of Stern's algorithm uses parameters p, ℓ, N, and N' whose size is determined in Section 7.

Stern's algorithm with overlapping sets. Let p be an integer with $0 \leq p \leq w$. Let ℓ be an integer with $0 \leq \ell \leq n-k$. Let N, N' be integers with $0 \leq N, N' \leq \binom{k}{p}$.

1. Choose an information set I.
2. Replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_I G_I^{-1} G$.
3. Select a uniform random size-ℓ subset Z in $\{1, \ldots, n\} \setminus I$.
4. Repeat N times: Choose a size-p subset $A = \{a_1, \ldots, a_p\} \subset I$ uniformly at random and consider the set $\mathcal{V}_A = \left\{\mathbf{y} - \sum_{i=1}^{p} m_i \mathbf{g}_{a_i} : \mathbf{m} = (m_1, \ldots, m_p) \in (\mathbf{F}_q^*)^p\right\}$. For each $\phi \in \mathcal{V}_A$ compute the vector $\phi(Z) \in \mathbf{F}_q^\ell$: the Z-indexed entries of ϕ.
5. Repeat N' times: Choose a size-p subset $B = \{b_1, \ldots, b_p\} \subset I$ uniformly at random and consider the set $\mathcal{V}_B = \left\{\sum_{j=1}^{p} m'_j \mathbf{g}_{b_j} : \mathbf{m}' = (m'_1, \ldots, m'_p) \in (\mathbf{F}_q^*)^p\right\}$. For each $\psi \in \mathcal{V}_B$ compute the vector $\psi(Z) \in \mathbf{F}_q^\ell$: the Z-indexed entries of ψ.
6. For each pair (A, B) where there is a pair of vectors ϕ and ψ such that $\phi(Z) = \psi(Z)$: Compute $\mathbf{e} = \mathbf{y} - \sum_i m_i \mathbf{g}_{a_i} - \sum_j m'_j \mathbf{g}_{b_j}$. If $\mathbf{e}$ has weight w print $\mathbf{e}$. Else go back to Step 1.

This algorithm finds a weight-w vector $\mathbf{e}$ in $\mathbf{y} + \mathbf{F}_q^k G$ if there is a vector $\mathbf{e}$ having weight $2p$ on positions indexed by the information set and weight 0 on the positions indexed by Z. Steps 1–6 form one iteration. If the set I chosen in Step 1 does not lead to a weight-w word in Step 6 another iteration has to be performed.

It is possible that a subset chosen in Step 4 is also chosen in Step 5. This case is allowed in order to benefit from a larger set of possible sums of p rows. The choice of N and N' is adjusted so that the number of overlapping sets is minimal.

7 Cost of Stern's algorithm with Finiasz–Sendrier's improvement

The analysis of the costs for the algorithm presented in Section 6 is done analogously to the analysis in Section 4 and Section 5.

In Step 4 one has to compute $N(q-1)^p$ vectors $\mathbf{y} - \sum_{i=1}^{p} m_i \mathbf{g}_{a_i}$ and in Step 5 one has to compute $N'(q-1)^p$ vectors — each vector on ℓ positions. First compute $k-p+1$ vectors $\mathbf{y} - \mathbf{g}_i$ with $i \in I$ and use intermediate sums so that each sum of p rows is computed using only one row addition, i.e., only ℓ additions in $\mathbf{F}_q$. The expected number of collisions in Step 6 is about $NN'(q-1)^{2p}/q^\ell$.

The total cost for one iteration of the algorithm is

$$\begin{aligned}
&(n-1)\left((k-1)\left(1-\frac{1}{q^r}\right)+(q^r-r)\right)\frac{c}{r}\\
&+\left((k-p+1)+(N+N')(q-1)^p\right)\ell\\
&+\frac{q}{q-1}(w-2p)2p\left(1+\frac{q-2}{q-1}\right)\frac{NN'(q-1)^{2p}}{q^\ell}.
\end{aligned}$$

Choice of parameters. As in Stern's algorithm the parameter p is chosen to be a small number. The parameter ℓ is classically chosen to balance the number of all computed length-ℓ vectors $\phi(Z)$ and $\psi(Z)$, $(N+N')(q-1)^p$, with the number of expected collisions on ℓ positions $NN'(q-1)^{2p}/q^\ell$. Assuming N and N' are about the same a reasonable choice is $\ell = \log_q N + p\log_q(q-1)$.

Determining N and N'. This algorithm works for any numbers N, N' less or equal to $\binom{k}{p}$, the number of all possible size-p subsets taken from an information set I. There is no point in choosing N larger than this number since otherwise all possible combinations of p elements out of I could be deterministically tested.

There are $\binom{2p}{p}$ different possibilities of splitting $2p$ errors into two disjoint subsets of cardinality p each. The probability of not finding an error vector $\mathbf{e}$, which has $2p$ errors in I, by a fixed set A and a fixed set B is $1-\binom{2p}{p}/\binom{k}{p}^2$. If one chooses N sets A and N' sets B uniformly at random the probability of $\mathbf{e}$ not being found by any pair (A,B) equals $\left(1-\binom{2p}{p}/\binom{k}{p}^2\right)^{NN'}$.

The probability of $\mathbf{e}$ not being found by sets A, B coming from random splits of $2p$ errors as p, p is

$$\left(1-\frac{\binom{2p}{p}}{\binom{k}{p}^2}\right)^{NN'} \approx \exp\left(-\frac{NN'\binom{2p}{p}}{\binom{k}{p}^2}\right).$$

Thus, a sensible choice for N and N' is $N = N' = \binom{k}{p}\Big/\sqrt{\binom{2p}{p}}$ since it minimizes the probability of $\mathbf{e}$ not being found by the algorithm.

The average number of ways to find $\mathbf{e}$ with $N = N' = \binom{k}{p}\Big/\sqrt{\binom{2p}{p}}$ is $\binom{k}{2p}\binom{n-k-\ell}{w-2p}\Big/\binom{n}{w}$.

8 Parameters

The iterations of Stern's algorithm for $\mathbf{F}_q$ with the speedups described in Section 4 are not independent. The number of iterations has to be estimated with a Markov chain computation as in [2, Section 5]. We adapted the Markov chain implementation from [2] to look for parameter ranges for McEliece-cryptosystem setups using codes over $\mathbf{F}_{31}$.

Our experiments show that an $[n, k]$-code with $n = 961$, $k = 771$, and $w = 48$ introduced errors over $\mathbf{F}_{31}$ achieves 128-bit security against the generalized Stern attack as presented in Section 3. Any Goppa code being the subfield subcode of a code in $\mathbf{F}_{31^2}$ with a degree-95 Goppa polynomial can be used. A successful attack needs about $2^{96.815}$ iterations with about $2^{32.165}$ bit operations per iteration and uses parameters $p = 2$, $\ell = 7$, $c = 12$, and $r = 1$. A public key for a $[961, 771]$ code over $\mathbf{F}_{31}$ would consist of $k(n-k)\log_2 31 = 725740$ bits.

For comparison: a $[2960, 2288]$ binary Goppa code where $w = 57$ errors are added by the sender has a public key of size 1537536 bits.

We are still working on obtaining similar results for $\mathbf{F}_3$ and $\mathbf{F}_4$ which will be included in future versions of this paper.

9 Outlook

We covered most of the improvements suggested in [2] but we omitted choosing multiple sets Z of size ℓ. This step amortizes the cost of the Gaussian elimination over more computations in the second part. For larger q the Gaussian elimination is even less of a bottleneck than in binary fields and we thus omitted this part here. A direction of future work is to review the exact cutoff between 2 and general q where this step is useful. We also plan to generalize further improvements due to Finiasz and Sendrier [5] — such as diagonalizing a smaller submatrix of the generator matrix — and investigate the extent to which they speed up the generalized algorithm.

References

1. Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386.
2. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography – Second International Workshop, PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, Berlin, 2008.
3. Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Henk C. A. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. In *Pre-proceedings of WCC 2009*, pages 168–180. 2009.
4. Anne Canteaut and Florent 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(1):367–378, 1998.
5. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Proceedings of AsiaCrypt 2009 (to appear)*. Springer.
6. Sean Hallgren and Ulrich Vollmer. Quantum computing. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 15–34. Springer, Berlin, 2008.
7. Gregory L. Katsman and Michael A. Tsfasman. A remark on algebraic geometric codes. In M. Isaacs, A. Lichtman, D. Passman, S. Sehgal, N. J. A. Sloane, and H. Zassenhaus, editors, *Representation theory, group rings, and coding theory*, volume 93, pages 197–199. American Mathematical Society, 1989.
8. Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In Christoph G. Günther, editor, *Advances in cryptology—EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, Berlin, 1988.
9. Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
10. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory, 1978. Jet Propulsion Laboratory DSN Progress Report 42–44. URL: `http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF`.

11. Rafael Misoczki and Paulo S. L. M. Barreto. Compact McEliece keys from Goppa codes. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 376–392, 2009.
12. Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 95–145. Springer, Berlin, 2008.
13. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, September 1962.
14. Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding theory and applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, New York, 1989.