An Attack on a Modified Niederreiter Encryption Scheme

Christian Wieschebrink

Federal Office for Information Security (BSI),
Godesberger Allee 185-189, 53175 Bonn, Germany
christian.wieschebrink@bsi.bund.de

Abstract. In [1] a Niederreiter-type public-key cryptosystem based on subcodes of generalized Reed-Solomon codes is presented. In this paper an algorithm is proposed which is able to recover the private key of the aforementioned system from the public key and which is considerably faster than a brute force attack. It is shown that the example parameters proposed in [1] are insecure.
Keywords. Public key cryptography, McEliece encryption, Niederreiter encryption, error-correcting codes, generalized Reed-Solomon codes, Sidelnikov-Shestakov attack

1 Introduction

The McEliece [2] and Niederreiter [3] encryption scheme are the most well-known code-based public key cryptosystems. Their security rests on two intractability assumptions: on the one hand it is difficult to decode an arbitrary linear code, on the other hand it is difficult to recover the structure of the underlying code from an arbitrary generator matrix which forms the public key in these systems. Indeed, the general syndrome decoding problem was shown in [4] to be NP-complete. Moreover there is practical evidence, that it is hard for random instances, too. Several quite sophisticated algorithms to attack the decoding problem were published (for example [5, 6]), but their running times remain exponential.

The hardness of the structural problem crucially depends on the kind of codes being used. The original Niederreiter scheme made use of generalized Reed-Solomon (GRS) codes. A polynomial time algorithm reconstrucing the code parameters from an arbitrary generator matrix was found afterwards by Sidelnikov and Shestakov [7]. Therefore the original Niederreiter scheme is completely broken. On the other hand McEliece proposed Goppa codes for his scheme. Up to now no efficient way is known to compute the parameters of these codes from the public key.

In [1] Berger and Loidreau propose a variant of the Niederreiter scheme which is intended to resist the Sidelnikov-Shestakov attack. The idea is to work with a subcode of a GRS code instead of a complete GRS code in order to hide its structure. In this paper we develop an attack on the modified system which is feasible if the subcode is chosen too large. It can be considered as a generalization of the Sidelnikov-Shestakov algorithm.

The rest of the article is structured as follows: after having presented the Berger-Loidreau variant in detail in Sect. 2 we describe the basic attack in Sect. 3. In Sect. 4 we show how to speed up this attack considerably and in Sect. 5 we give some results of a test implementation.

2 The Modified Scheme

First of all let's recall some basic facts about generalized Reed-Solomon codes. In the following let F be a finite field.

Definition 1. *Let $m, k, n \in \mathbb{N}$, $k \leq n$, $\alpha = (\alpha_1, \dots, \alpha_n) \in F^n$, $x = (x_1, \dots, x_n) \in (F \backslash \{0\})^n$, where the α_i are pairwise distinct. The* generalized Reed-Solomon code (or GRS code) *$GRS_{n,k}(\alpha, x)$ is a linear code over F given by the generator matrix*

$$G_{\alpha,x} = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_1\alpha_1 & x_2\alpha_2 & \dots & x_n\alpha_n \\ \vdots & & \ddots & \\ x_1\alpha_1^{k-1} & x_2\alpha_2^{k-1} & \cdots & x_n\alpha_n^{k-1} \end{pmatrix} .$$

Consequently $GRS_{n,k}(\alpha, x)$ consists exactly of those words c in F^n which can be written $c = (x_1 f(\alpha_1), \dots, x_n f(\alpha_n))$ for a polynomial $f(x) \in F[x]$ with deg f $< k$. GRS codes allow efficient error correction. Given x and α one can apply the Berlekamp-Massey algorithm which can correct up to $\lfloor \frac{n-k}{2} \rfloor$ errors in polynomial time. (For details see [8, 9].) In context of cryptography it is always assumed that $GRS_{n,k}(\alpha, x)$ has full length, i.e. $n = \#F$ and char $F = 2$. For a fixed GRS code $GRS_{n,k}(\alpha, x)$ the parameters α and x are not uniquely determined:

Proposition 1. *Let α, x be defined as above. Then*

$$GRS_{n,k}(\alpha, x) = GRS_{n,k}((a\alpha_1 + b, \dots, a\alpha_n + b), (cx_1, \dots, cx_n))$$

for all $a, b, c \in F$, $a, c \neq 0$.

Proof. See [9]. □

It follows for example that two of the α_i can be chosen arbitrarily. Each of the different parameters for a given GRS code is equally suited for the above mentioned decoding algorithm.

Proposition 2. *Let α, x be defined as above and $u := (u_1, \dots u_n)$ where $u_i := x_i^{-1} \prod_{j \neq i} (\alpha_i - \alpha_j)^{-1}$. Then the dual code of $GRS_{n,k}(\alpha, x)$ is given by*

$$GRS_{n,k}(\alpha, x)^{\perp} = GRS_{n,n-k}(\alpha, u) .$$

Proof. See [9]. □

Proposition 2 will be helpful later for reconstructing x if α is known.

The Berger-Loidreau modification of the Niederreiter public-key scheme works as follows (we present the dual version of the scheme given in [1], which has the same security, see [10]):

Key creation: Let $n = \#F$, $k \in \mathbb{N}^{\leq n}$ and a small $l \in \mathbb{N}^{\leq k}$ be given. Alice chooses a random GRS code $GRS_{n,k}(\alpha, x)$ with generator matrix $G_{\alpha,x}$ and a random $(k-l) \times k-$matrix A over F of rank $k-l$. Then her public key is given by $T := A \cdot G_{\alpha,x}$. The secret key is (α, x). (A must be kept secret, too.)

Encryption: To encrypt a message $m \in F^{k-l}$ Bob chooses a (secret) vector $e \in F^n$ of Hamming weight $\leq \lfloor \frac{n-k}{2} \rfloor$ and computes the cipertext $c := mT + e$.

Decryption: Using (α, x) Alice applies the decoding algorithm to c getting mT. By multiplying this with a right-side inverse of T she gets m.

3 The Attack

We fix some additional notation. For a $(k \times n)-$matrix $T = (t_{i,j})$ let $E(T)$ be the echelon form of T and $<T>$ the code generated by T. The $i-$th row of T is denoted by t_i. Given a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ let T_π denote the matrix $(t_{i,\pi(j)})$, i.e. the columns of T are permuted according to π. Analogically for $v = (v_1, \ldots v_n) \in F^n$ we define $v_\pi := (v_{\pi(1)}, \ldots v_{\pi(n)})$. If T is a generator matrix of $GRS_{n,k}(\alpha, x)$ then obviously T_π is a generator matrix of $GRS_{n,k}(\alpha_\pi, x_\pi)$.

Now let T be the public key of the aforementioned encryption scheme. Clearly T is a generator matrix of a $(k-l)-$dimensional subcode of $GRS_{n,k}(\alpha, x)$. Our aim is to find the parameters α and x (or equivalent parameters, see Proposition 1) where only T is given. The attack consists of two steps. In the first step (which is the more expensive one) the permutation of the field elements α is calculated. In the second step x is recovered.

Let $c \in <T>$. Recall that c can be written in the form

$$c = (x_1 f(\alpha_1), \ldots, x_n f(\alpha_n)) \ , \tag{1}$$

where $f \in F[x]$ with deg $f \leq k-1$. Now let $d \in <T>$ be another codeword, $d = (x_1 g(\alpha_1), \ldots, x_n g(\alpha_n))$. For all $i = 1, \ldots, n$ we then have

$$\frac{c_i}{d_i} = \frac{x_i f(\alpha_i)}{x_i g(\alpha_i)} = \frac{f}{g}(\alpha_i) \ ,$$

unless $d_i = 0$. The main idea of the attack is based on the following

Proposition 3. *Let T be the generator matrix of a $(k-l)$-dimensional subcode of $GRS_{n,k}(\alpha, x)$ and $E(T) := (t_{i,j}) = [1_{k-l}|A]$ the echelon form of T. Then for each pair $(i, b) \in \{1, \ldots, k-l\}^2$ there are polynomials $P_i(x), P_b(x) \in F[x]$ of degree $\leq l$ such that*

$$\frac{t_{i,j}}{t_{b,j}} = \frac{(\alpha_j - \alpha_b) P_i(\alpha_j)}{(\alpha_j - \alpha_i) P_b(\alpha_j)} \tag{2}$$

for all $j = 1, \ldots n$ with $t_{b,j} \neq 0$.

Proof. For given i, b let t_i, t_b the respective rows of $E(T)$. Since $E(T)$ is in echelon form both rows contain (at least) $k-l-1$ zeros, and there are (at least) $k-l-2$ positions where t_i, t_b have common zeros. Let $a_1, \ldots, a_{k-l-2} \in \{1, \ldots, k-l\}$ be these positions. According to the properties of a GRS code there are polynomials $f_i(x), f_b(x) \in F[x]$ of degree $\leq k-1$, s.t.

$$(t_{c,1}, \ldots, t_{c,n}) = (x_1 f_c(\alpha_1), \ldots, x_n f_c(\alpha_n))$$

for $c = i, b$ and they must have the form

$$f_i(x) = (x - \alpha_b) \cdot P_i(x) \cdot \prod_{r=1}^{k-l-2} (x - \alpha_{a_r}) \ ,$$

$$f_b(x) = (x - \alpha_i) \cdot P_b(x) \cdot \prod_{r=1}^{k-l-2} (x - \alpha_{a_r})$$

with P_i and P_b having degree l at most. So for all $j = 1, \ldots, n$ with $t_{b,j} \neq 0$ we have

$$\frac{t_{i,j}}{t_{b,j}} = \frac{f_i(\alpha_j)}{f_b(\alpha_j)} = \frac{(\alpha_j - \alpha_b) P_i(\alpha_j)}{(\alpha_j - \alpha_i) P_b(\alpha_j)} \ .$$

□

Note that P_i, P_b in the above proposition may have common factors, so these polynomials are not unique in general. Since P_i and P_b have low degree we can now try to reconstruct the coefficients of both polynomials. If we do so for different rows t_i of $E(T)$ it is possible to recover the α_i as we will see below.

First of all we need a simple

Lemma 1. *Let $f(x) = \frac{P(x)}{Q(x)}$ be a rational function over F with deg $P, Q \leq i \in \mathbb{N}$, P, Q relatively prime and Q monic. Let $x_1, \ldots, x_{2i+1} \in F$ be pairwise distinct values, for which f is defined. Then the coefficients of P and Q are uniquely determined by the pairs $(x_j, f(x_j)), j = 1, \ldots, 2i+1$ and can be computed in polynomial time.*

Proof. Let $\bar{P}, \bar{Q}$ be another pair of relatively prime polynomials over F with $f(x) = \frac{\bar{P}(x)}{\bar{Q}(x)}$, deg $\bar{P}, \bar{Q} \leq i$ and $\bar{Q}$ monic. Then we have $P(x_j)\bar{Q}(x_j) = \bar{P}(x_j)Q(x_j)$ for $j = 1, \ldots, 2i+1$. Since $P\bar{Q}$ and $\bar{P}Q$ are polynomials of degree $\leq 2i$ it follows $P\bar{Q} = \bar{P}Q$. According to our assumptions P and Q have no common divisions, so we have $Q|\bar{Q}$ and analogically $\bar{Q}|Q$. $\bar{Q}$ and Q are monic, so $\bar{Q} = Q$. It follows $\bar{P} = P$ immediately. This shows the uniqueness of P and Q.

Now let $P(x) = p_i x^i + \cdots + p_1 x + p_0$, $Q(x) = q_i x^i + \cdots + q_1 x + q_0$. As the $f(x_j)$ are defined, we get

$$f(x_j) q_i x_j^i + \cdots + f(x_j) q_1 x_j + f(x_j) q_0 - p_i x_j^i - \cdots - p_1 x_j - p_0 = 0$$

for $j = 1, \ldots, 2i+1$. This yields a (inhomogenous) linear system in the unknowns $q_i, \ldots q_0, p_i, \ldots, p_0$, which can be solved with $O(i^3)$ operations in F. The solution space may have dimension $d > 1$. In this case the unique solution polynomials P, Q in the above sense have both degree less than i. To find them one has to compute the element of the solution space with $q_i = q_{i-1} = \cdots = q_{i-(d-2)} = 0$, $q_{i-(d-1)} = 1$. Obviously this can be done in polynomial time, too. $\square$

Now consider (2) again. We're fixing an arbitrary b, for example $b = k-l =: r$, and put

$$\tilde{P}_i(x) := (x - \alpha_r)P_i(x), \tilde{Q}_i(x) := (x - \alpha_i)P_r(x) \tag{3}$$

and $g_i(x) := \frac{\tilde{P}_i(x)}{\tilde{Q}_i(x)}$ for $i = 1, \dots, r-1$. The first step is to reconstruct the coefficients of $\tilde{P}_i$ and $\tilde{Q}_i$. These polynomials have degree $\leq l+1$ so according to the lemma above we need to know $2l+3$ pairs $(\alpha_j, g_i(\alpha_j))$ to do so. The $g_i(\alpha_j) = \frac{t_{i,j}}{t_{r,j}}$ are given, but the α_j are unknown. The strategy is now to guess the values $\alpha_{r+1}, \dots, \alpha_{r+2l+3}$ (for example) and sieve out the wrong guesses. W.l.o.g. we assume that $t_{r,r+1}, \dots, t_{r,r+2l+3}$ all are nonzero (otherwise we can choose a different set of $2l+3$ indices $i_1, \dots, i_{2l+3} \in \{r+1, \dots, n\}$ with $t_{i_1}, \dots, t_{i_{2l+3}} \neq 0$ and guess the values $\alpha_{i_1}, \dots, \alpha_{i_{2l+3}}$) and that $\alpha_{r+1} = 0$, $\alpha_{r+2} = 1$ (by Proposition 1), s.t. in fact only $\alpha_{r+3}, \dots, \alpha_{r+2l+3}$ have to be guessed. Given the pairs $(\alpha_j, \frac{t_{i,j}}{t_{r,j}})$ we calculate relatively prime P_i^*, Q_i^* of degree $\leq l+1$ with $\frac{P_i^*}{Q_i^*} = g_i$ for $i = 1, \dots, r-1$ by solving the appropriate linear systems, see Lemma 1. Note that $\tilde{P}_i$ and $\tilde{Q}_i$ may have a nontrivial common factor, so in general $\tilde{P}_i \neq P_i^*$ and $\tilde{Q}_i \neq Q_i^*$. However, if the guess was correct then the following conditions hold:

C1. The $P_i^*(x)$ have a common linear factor (namely $(x - \alpha_r)$).
C2. There is a sequence $\alpha_1, \dots, \alpha_{r-1}$ of pairwise distinct elements of F different from the $\alpha_{r+1}, \dots, \alpha_{r+2l+3}$, such that $(x - \alpha_i)$ divides $Q_i^*(x)$, and the least common multiple of the $\frac{Q_i^*(x)}{(x-\alpha_i)}$ has degree $\leq l$ (the least common multiple divides $P_r(x)$).

If there are two distinct polynomials P_i^*, P_j^* with degree $l+1$ then $Q_i^* = \tilde{Q}_i$ and $Q_j^* = \tilde{Q}_j$ (assuming that these polynomials are monic), and condition C2 can be replaced by

C3. Let $Q := gcd(Q_i^*, Q_j^*)$. Then $\frac{Q_w^*(x)}{gcd(Q_w^*(x), Q(x))} = (x - \alpha_w)$, $w = 1, \dots, r-1$ for pairwise distinct $\alpha_1, \dots, \alpha_{r-1}$ different from $\alpha_{r+1}, \dots, \alpha_{r+2l+3}$ (it is $Q = P_r$).

The advantage of C3 is that it is straightforward to check from an algorithmic point of view, while C2 is more complicated (but also can be checked in polynomial time). So we always assume first that there is such a pair P_i^*, P_j^*, which is the case with high probability. Condition C1 can be verified easily, too, by the Euclidian algorithm. If the guess was right we can reconstruct the parameter $\alpha = (\alpha_1, \dots, \alpha_n)$ of the GRS code from the P_i^*, Q_i^*: α_r can be reconstructed from condition C1 and the values $\alpha_1, \dots, \alpha_{r-1}$ can be derived from condition C3. $\alpha_{r+1}, \dots, \alpha_{r+2l+3}$ are given so it remains to find the $\alpha_{r+2l+4}, \dots, \alpha_n$.

Suppose $\alpha_{i_1}, \dots, \alpha_{i_{r-1}}$ belong to the unknown values. Choose a permutation $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ with $\pi(j) = i_j$ and $\pi(i_j) = j$ for $j = 1, \dots, r-1$ and

$\pi(b) = b$ for $b = r+1, \dots, r+2l+3$. Let $\beta := (\beta_1, \dots, \beta_n) := (\alpha_{\pi(1)}, \dots, \alpha_{\pi(n)})$. The matrix T_π is a generator matrix of a subcode of $GRS_{n,k}(\beta, x_\pi)$. Since the $\beta_i = \alpha_i$, $i = r+1, \dots r+2l+3$ are given, the $\beta_1, \dots \beta_{r-1}$ – and thereby the $\alpha_{i_1}, \dots, \alpha_{i_{r-1}}$– can be determined exactly the same way as described above when working with $E(T_\pi)$ instead of $E(T)$. This process can be repeated for different suitable permutations until all α_i are found.

We summarize the complete procedure in Algorithm 1. It makes use of the function *getAlpha* which is defined in Algorithm 2.

Algorithm 1 Reconstruction of α

Input: Generator matrix T of a subcode of $GRS_{n,k}(\alpha, x)$ of dimension $r = k - l$
Output: Set B of candidates for α

$B \leftarrow \emptyset$
$\beta_1 \leftarrow 0$
$\beta_2 \leftarrow 1$
for all $(\beta_3, \dots, \beta_{2l+3}) \in (F \backslash \{0,1\})^{2l+1}$ with β_i pairwise distinct **do**
  $I \leftarrow \{1, \dots, r-1, r, r+2l+4, \dots, n\}$
  repeat
    $b \leftarrow \min(r-1, \#I)$
    for $j \leftarrow 1, \dots, b$ **do**
      $i_j \leftarrow$ least element of I
      $I \leftarrow I \backslash \{i_j\}$
    end for
    for $j \leftarrow 1, \dots, b$ **do**
      $\pi(j) \leftarrow i_j$
      $\pi(i_j) \leftarrow j$
    end for
    for $j \leftarrow b+1, \dots, n$ **do**
      if $j \neq i_1, \dots, i_b$ **then**
        $\pi(j) \leftarrow j$
      end if
    end for
    calculate T_π
    $\gamma := (\gamma_1, \dots, \gamma_{r-1}) \leftarrow getAlpha(\beta_1, \dots, \beta_{2l+3}, T_\pi)$
    if $\gamma \neq NULL$ **then**
      for $j \leftarrow 1, \dots, b$ **do**
        $\alpha_{i_j} \leftarrow \gamma_j$
      end for
    end if
  until $I = \emptyset$ or $\gamma = NULL$
  if $\gamma \neq NULL$ and $\alpha_1, \dots, \alpha_n$ pairwise distinct **then**
    $B \leftarrow B \cup \{(\alpha_1, \dots, \alpha_n)\}$
  end if
end for
return B

Algorithm 2 $getAlpha(\beta_1, \ldots, \beta_{2l+3}, T)$

Input: $(r \times n)$-matrix T over F, $\beta_1, \ldots, \beta_{2l+3} \in F$ pairwise distinct
Output: $(\alpha_1, \ldots, \alpha_{r-1}) \in F^{r-1}$

1: $(t_{i,j}) \leftarrow$ echelon form of T
2: **for** $i \leftarrow 1, \ldots, r-1$ **do**
3: calculate relatively prime $P_i^*(x), Q_i^*(x) \in F[x]$ with degree $\leq l+1$ and Q_i^* monic and

$$\frac{P_i^*(\beta_j)}{Q_i^*(\beta_j)} = \frac{t_{i,r+j}}{t_{r,r+j}}$$

for all $j = 1, \ldots, 2l+3$
4: **end for**
5: **if** the $P_i^*(x), Q_i^*(x)$ satisfy conditions C1 and C3 **then**
6: $Q \leftarrow gcd(Q_i^*, Q_j^*)$ with i, j such that $i \neq j$ and deg $P_i^* =$ deg $P_j^* = l+1$
7: **for** $i \leftarrow 1, \ldots, r-1$ **do**
8: $\alpha_i \leftarrow root(\frac{Q_i^*}{gcd(Q_i^*, Q)})$
9: **end for**
10: **return** $(\alpha_1, \ldots, \alpha_{r-1})$
11: **else**
12: **return** $NULL$
13: **end if**

Once the set of candidates B is given is remains to check for each $\alpha' \in B$ if there is a $x = (x_1, \ldots x_n)$, s.t. $<T> \subset GRS_{n,k}(\alpha', x)$. (We know that there is at least one such α'.) This can be done using Algorithm 3.

According to Proposition 2 the dual code of $GRS_{n,k}(\alpha, x)$ is also a GRS code $G = GRS_{n,n-k}(\alpha, x')$. Let g be a row of the canonical generator matrix of G. Since each row vector t of T is an element of $GRS_{n,k}(\alpha, x)$ the inner product $t \cdot g$ is equal to zero. That's why $x' = (x'_1, \ldots, x'_n)$ has to be a solution of the linear system

$$t_{i,1}\alpha_1^j x'_1 + \ldots + t_{i,n}\alpha_n^j x'_n = 0, \quad i = 1, \ldots, r, \quad j = 0, \ldots, n-k-1 \ .$$

If such a x' is found, the vector x can be calculated with help of Proposition 2.

Let's analyze the running time of the above algorithms in the worst case. First consider the function $getAlpha$. It is dominated by the computation of the echelon form in line 1, which takes $O(r^2 n)$ operations in F, and the for-loop in lines 3–4. In each step of the loop a linear system with $O(l)$ equations and unknowns has to be solved, which can be done with $O(l^3)$ operations. Verification of C1 and C3 and computation of the α_i takes $O(rl^2)$ operations at most. This yields a total running time of $O(r^2 n + rl^3)$ for Algorithm 2.

The main loop in lines 4–32 of Algorithm 1 is run $\frac{(n-2)!}{(n-2l-3)!} \in O(n^{2l+1})$ times. (We assumed $n = \#F$). The inner loop in lines 6–28 is called $\left\lceil \frac{n-2l-3}{r-1} \right\rceil$ times. Since in practice $\frac{n}{3} \leq r \leq \frac{2n}{3}$ we can assume that this value is bounded by a constant. With the above result we get a total running time of $O(n^{2l+1}(r^2 n + rl^3))$

Algorithm 3 Reconstruction of x

Input: $T = (v_{i,j}), B$ as in Algorithm 1
Output: (α, x') s.t. $<T>\subset GRS_{n,k}(\alpha, x')$

while $B \neq \emptyset$ **do**
  $(\alpha_1, \ldots, \alpha_n) \leftarrow$ arbitrary element of B
  $X \leftarrow$ solution space of the linear system in $x_1, \ldots, x_n$ given by

$$v_{m,1}\alpha_1 x_1^j + v_{m,2}\alpha_2 x_2^j + \cdots + v_{m,n}\alpha_n x_n^j = 0$$

  for $j = 0, \ldots, k-1$ and $m = 1, \ldots, r$
  if $dim(X) > 0$ **then**
    $(x_1, \ldots, x_n) \leftarrow$ arbitrary nonzero element of X
    for $i \leftarrow 1, \ldots, n$ **do**
      $x'_i \leftarrow (x_i \prod_{j \neq i}(\alpha_i - \alpha_j))^{-1}$
    end for
    $B \leftarrow \emptyset$
  else
    $B \leftarrow B \backslash \{(\alpha_1, \ldots, \alpha_n)\}$
  end if
end while
return $((\alpha_1, \ldots, \alpha_n), (x'_1, \ldots, x'_n))$

operations in F. Note that the procedure can be optimized by computing the echelon forms $E(T_\pi)$ for a fixed set of suitable permutations π in advance instead of computing them in each call of *getAlpha*. In this case we get an upper bound $O(r^2 n + n^{2l+1} r l^3)$.

In Algorithm 3 the main loop is run $\#B$ times, and the dominant step in each loop is the linear system. It has n unknowns and $(k-1)r$ equations so it takes at most $O(n^2 kr)$ operations to find a nontrivial solution. We get a worst case complexity of $O(\#B \cdot n^2 kr)$ operations. In general $\#B$ is expected to be quite small so that Algorithm 3 is feasible.

In [1] an attack on the cryptosystem is given, which uses the original Sidelnikov-Shestakov attack as a black box algorithm. Its average running time is lower bounded by $\Omega(n^{kl})$ operations, so for practical choices of n, k, l the attack given here is much faster.

4 Refinement of the Attack

The above algorithm can be improved if there are two rows in the echelon form $E(T) = (t_{i,j})$ which have more than $k - l - 2$ zeros in common. Suppose the i-th and the b-th row, $i \neq b$, have $k - l - 2 + s$ zeros in common positions. It is $0 \leq s \leq l$. With the same argument as in proof of Proposition 3 there are two polynomials $P^*(x), Q^*(x) \in F[x]$ of degree $\leq l - s + 1$ (instead of $l + 1$) s.t.

$$\frac{t_{i,j}}{t_{b,j}} = \frac{P^*(\alpha_j)}{Q^*(\alpha_j)}$$

for all $j = 1, \dots, n$ with $t_{b,j} \neq 0$. So to find these polynomials only $2(l-s+2)-1 = 2(l-s)+3$ of the α_j have to be known according to Lemma 1, and the number of guesses which have to be made is reduced by a factor $O(n^{2s})$. To check whether the guess is correct we make use of the following

Definition 2. *Let S be a (finite) set, $n, k \in \mathbb{N}, n \geq k$ and $v = (v_1, \dots, v_n) \in S^n$,$w = (w_1, \dots, w_k) \in S^k$. We say that v dominates w, if*

$$\#\{i | v_i = s\} \geq \#\{j | w_j = s\}$$

for all $s \in S$.

Obviously for given $v \in S^n, w \in S^k$ it can be checked with $O(n)$ operations if v dominates w.

Let $J \subset \{1, \dots, n\}$ be the set of those j, where $t_{i,j} \neq 0$ or $t_{b,j} \neq 0$. For $\gamma \in F$ with $\gamma \neq 0$ we define $\frac{\gamma}{0} =: \infty$. Suppose the elements of F are ordered in some way. If the guess of the α_j is correct then the vector $(\frac{P^*(\gamma)}{Q^*(\gamma)})_{\gamma \in F}$ has to dominate the vector $(\frac{t_{i,j}}{t_{b,j}})_{j \in J}$. In this case it may be possible to reconstruct some of the (not yet assigned) α_j : suppose the function $f(x) := \frac{P^*(x)}{Q^*(x)}$ takes the value $\delta \in F \cup \{\infty\}$ for exactly one $\gamma \in F$, $f(\gamma) = \delta$, and there is a $j \in J$ with $\frac{t_{i,j}}{t_{b,j}} = \delta$. Then $\alpha_j = \gamma$. If we can find at least $2s$ additional α_j with $t_{b,j} \neq 0$ this way we can try to compute relatively prime polynomials $P^*_{i'}(x)$,$Q^*_{i'}(x) \in F[x]$ of degree $\leq l+1$ for $i' \in \{1, \dots k-l\} \backslash \{i, b\}$ with

$$\frac{t_{i',j}}{t_{b,j}} = \frac{P^*_{i'}(\alpha_j)}{Q^*_{i'}(\alpha_j)} \tag{4}$$

for all $j = 1, \dots, n$ with $t_{b,j} \neq 0$. Of course the right polynomials have to comply with conditions C1 and C2 / C3. This allows us to reconstruct the remaining α_j as seen above.

If there are not enough δ s.t. $f(\gamma) = \delta$ can be solved uniquely, then at least we can extract a list of candidates for each α_j, $j \in J$, which consists of all γ with $f(\gamma) = \frac{t_{i,j}}{t_{b,j}}$. We can then choose a sufficient number of short candidate lists and try to solve (4) with the different possible assignments for the α_j.

What can we do now, if a pair of rows in $E(T)$ with more than $k-l-2$ common zeros does not exist? In this case we can try to find such a pair in the echelon form of an equivalent code of $<T> \subset GRS_{n,k}(\alpha, x)$. Let $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ be a permutation. Remember that due to the definition of GRS codes the matrix T_π is a generator matrix of a subcode of $GRS_{n,k}(\alpha_\pi, x_\pi)$. So we can replace T by T_π for distinct permutations π and look for rows in the echelon form $E(T_\pi)$ which have more than $k-l-2$ common zeros. When such a pair is found we apply the above method which eventually finds a set of candidates for α_π, which can easily be transformed to a set of candidates for α. When choosing the permutations we can restrict ourselves to those π which satisfy $\pi(i) > k-l$ for at least one $i \in \{1, \dots, k-l\}$, since otherwise $E(T_\pi)$ differs from $E(T)$ only by the order of rows.

Note however that such a pair of rows does not necessarily exist in any equivalent code. For example the subcode C can itself be a GRS code of dimension $k-l$. As such it is a MDS code and any pair of rows in the echelon form can have $k-l-2$ common zeros at most. But for random instances there should be a good chance of finding a pair at least for small s.

The improved approach is summarized in Algorithms 4 and 5.

We try to give a rough estimate for the running time of Algorithm 5. The main loop in lines 12–30 is run $O(n^{2(l-s)+1})$ times. Solving the linear system in line 13 takes $O((l-s)^3)$ operations. If the condition in line 15 is passed (verification takes $O(n(l-s))$ operations) the for-loop in lines 21–28 is called $O(\max_j\{B_j\}^{2s})$ times at most. Each loop takes $O(r^2n+rl^3)$ operations. Since $\max_j\{B_j\} \leq l-s+1$ we get an upper bound $O(t_1+n^{2(l-s)+1}((l-s)^3+n(l-s)+(l-s+1)^{2s}(r^2n+rl^3)))$ for the complete algorithm, where t_1 is the (undetermined) running time of Algorithm 4. Here we assumed that the condition in line 15 is always passed, which won't be the case in practice. The average running time should be well below the given bound.

Note that there are still several possibilities to improve the presented algorithms but for the sake of clarity we didn't include them here.

Algorithm 4 $findPermutation(T, s)$

Input: $(r \times n)$-matrix T as in Algorithm 1; $s \in \mathbb{N}^{\leq l}$
Output: (π, i, b) s.t. i-th and b-th row of $E(T_\pi)$ have $r+s-2$ common zeros

```
S ← set of all permutations π ∈ S_n with π(i) > r for some i ∈ N^{≤r}
repeat
    π ← random element of S
    S ← S\{π}
    calculate E(T_π)
    for all (i, b) ∈ {1, ..., r}^2 with i < b do
        if rows i and b of E(T_π) have r + s − 2 common zeros then
            return (π, i, b)
        end if
    end for
until S = ∅
return NULL
```

5 Experimental Results

Algorithms 4 and 5 were implemented in JAVA (with some minor modifications) and executed for different instances of the encryption scheme. We always chose s such that $l-s=1$. Table 1 shows some example running times on a 2.6 GHz Pentium 4, 512 MB system. In particular we see that $findPemutation$ performs well for small s.

Algorithm 5 Reconstruction of α, improved version

Input: Generator matrix T of a subcode of $GRS_{n,k}(\alpha, x)$ of dimension $r = k - l$, $s \in \mathbb{N}^{\leq l}$

Output: Set B of candidates for α

$(\pi, i, b) \leftarrow findPermutation(T, s)$
if $(\pi, i, b) = NULL$ **then**
  return $NULL$
end if
compute $(t_{i,j}) := E(T_\pi)$
$(a_1, \ldots, a_n) \leftarrow i-$th row of $E(T_\pi)$
$(b_1, \ldots, b_n) \leftarrow b-$th row of $E(T_\pi)$
$B \leftarrow \emptyset$
$\beta_1 \leftarrow 0$
$\beta_2 \leftarrow 1$
find pairwise distinct $i_1, \ldots, i_{2(l-s)+3}$ s.t. $b_{i_j} \neq 0$ for all j
for all $(\beta_3, \ldots, \beta_{2(l-s)+3}) \in (F \backslash \{0, 1\})^2$ with β_i pairwise distinct **do**
  compute relatively prime $P^*(x), Q^*(x) \in F[x]$ with deg $P^*(x), Q^*(x) \leq l-s+1$ s.t.

$$\frac{P^*(\beta_j)}{Q^*(\beta_j)} = \frac{a_{i_j}}{b_{i_j}}$$

  for all $j = 1, \ldots, 2(l-s)+3$
  $c \leftarrow (\frac{a_i}{b_i})_{i \in \{1,\ldots,n\}, a_i \neq 0 \ or \ b_i \neq 0}$
  if $(\frac{P^*(\gamma)}{Q^*(\gamma)})_{\gamma \in F}$ dominates c **then**
    $I \leftarrow \{r+1, \ldots, n\} \backslash \{i_1, \ldots, i_{2(l-s)+1}\}$
    find pairwise distinct $i_{2(l-s)+4}, \ldots, i_{2l+3} \in I$ with $b_{i_j} \neq 0$
    for $j \leftarrow 2(l-s)+4, \ldots, 2l+3$ **do**
      $B_j \leftarrow$ set of all $\gamma \in F \backslash \{\beta_1, \ldots, \beta_{2(l-s)+3}\}$ with $\frac{P^*(\gamma)}{Q^*(\gamma)} = \frac{a_{i_j}}{b_{i_j}}$
    end for
    for all $(\beta_{2(l-s)+4}, \ldots, \beta_{2l+3})$ with pairw. distinct $\beta_j \in B_j$ **do**
      for all $a \in \{1, \ldots, r-1\} \backslash \{b\}$ **do**
        compute relatively prime $P_i^*(x), Q_i^*(x) \in F[x]$ with degree $\leq l+1$ s.t.

$$\frac{P_i^*(x)}{Q_i^*(x)} = \frac{t_{a,i_j}}{b_{i_j}}$$

        for all $j = 1, \ldots, 2l+3$
      end for
      if the P_i^*, Q_i^* suffice conditions C1 and C2/C3 and α_π can be computed as in Algorithms 1,2 **then**
        $B \leftarrow B \cup \{\alpha\}$
      end if
    end for
  end if
end for
return B

Table 1. Perfomance for different key parameters

				running time	
n	k	l	s	$findPermutation$	total
32	16	3	2	3 sec	<1 min
64	32	3	2	2 sec	16 min
64	40	3	2	2 sec	16 min
64	32	4	3	2 min	18 min
128	64	4	3	20 min	5 h 44 min

6 Conclusion

In [1] the values $n = 256$, $k = 133$, $l = 4$ are given as secure example parameters for the modified Niederreiter encryption scheme. It is claimed that $\approx 2^{2000}$ executions of the Sidelnikov-Shestakov algorithm for a structural break are needed in a brute force approach. However the above results suggest that these choices for the modified Niederreiter encryption scheme are highly insecure. Extrapolating the data above we estimate that an optimized implementation of the above attack can break such a system in a few days or even hours on a PC.

The encryption scheme is not completely broken though. To thwart the attack n and l should be chosen sufficiently large. However this has other drawbacks. A large n leads to large public keys and a large l causes bigger message expansion. It is unclear if the parameters can be chosen in such a way that it has higher efficiency and security than the McEliece cryptosystem.

References

1. Berger, T., Loidreau, P.: How to mask the structure of codes for a cryptographic use. Designs, Codes and Cryptography **35**(1) (2005) 63–79
2. McEliece, R.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Prop. Lab., California Inst. Tech. **42-44** (1978) 114–116
3. Niederreiter, N.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory **15** (1986) 159–166
4. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory **24**(3) (1978) 384–386
5. Brickell, E., Lee, J.: An observation on the security of McEliece's public-key cryptosystem. In: EUROCRYPT '88. Number 330 in Lecture Notes in Computer Science, Springer-Verlag (1988) 275–280
6. Canteaut, A., Chabaud, F.: 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) (1988) 367–378
7. Sidelnikov, V., Shestakov, S.: On insecurity of cryptosystems based on generalized Reed-Solomon codes. Discrete Math. Appl. **2**(4) (1992) 439–444

8. Gabidulin, E.: Public-key cryptosystems based on linear codes (1995) http://citeseer.ist.psu.edu/gabidulin95publickey.html.
9. MacWilliams, F., Sloane, N.: The Theory of Error-Correcting Codes. North Holland (1997)
10. Deng, R., Li, Y., Wang, X.: On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Transactions on Information Theory **40**(1) (1994) 271–273
11. Garey, M., Johnson, D.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Company (1979)
12. Overbeck, R.: A new structural attack for GPT and variants. In: Mycrypt 2005. Number 3715 in Lecture Notes in Computer Science, Springer-Verlag (2005)