A Welch–Berlekamp Like Algorithm for Decoding Gabidulin Codes

Pierre Loidreau

Ecole Nationale Supérieure de Techniques Avancées
Pierre.Loidreau@ensta.fr

Abstract. In this paper, we present a new approach of the decoding of Gabidulin codes. We show that, in the same way as decoding Reed-Solomon codes is an instance of the problem called *polynomial reconstruction*, the decoding of Gabidulin codes can be seen as an instance of the problem of reconstruction of linearized polynomials. This approach leads to the design of two efficient decoding algorithms inspired from the Welch–Berlekamp decoding algorithm for Reed–Solomon codes. The first algorithm has the same complexity as the existing ones, that is cubic in the number of errors, whereas the second has quadratic complexity in $2.5n^2 - 1.5k^2$.

1 Introduction

Gabidulin codes are the analogs for rank metric of Reed–Solomon codes for Hamming metric. Namely, they consist of evaluation of q–polynomials of bounded degree over a set of elements of a finite field, [3]. These codes are optimal codes, both in Hamming and in rank metric and can be used in building cryptosystems, with a much smaller public-key size than McEliece type cryptosystems whose security relies on the difficulty of decoding in Hamming metric [5]. Several polynomial-time decoding algorithms were designed until now enabling to decode Gabidulin codes up to their rank error-correcting capability. It is interesting to note that all of them have an equivalent decoding algorithm in Hamming metric for Reed–Solomon codes, such as *extended Euclidian*, and *Berlekamp–Massey* algorithms, [3,4,11,10].

Concerning Reed-Solomon codes there is still another decoding algorithm based on the analogy between decoding Reed–Solomon codes and solving some instances of the polynomial reconstruction problem [12]. Inspired by such an analogy we reformulated the problem of decoding Gabidulin codes into the problem of *q–polynomial reconstruction.* In the following, we show that the problem of decoding Gabidulin codes can be related to this problem in a simple way. We then derive two polynomial-time decoding algorithms solving this problem. They can be seen as the analogs in rank metric of Welch–Berlekamp algorithms, [1].

2 Rank Metric and Gabidulin Codes

Rank metric was introduced in 1985 by E.M. Gabidulin [3]. Given a vector $\mathbf{c} = (c_1, \ldots, c_n)$ of elements of a finite field $GF(q^m)$, the rank over $GF(q)$ of $\mathbf{c}$

is defined as the rank of the $n \times m$ q-ary matrix obtained by expanding each coordinate of $\mathbf{c}$ over a basis of $GF(q^m)/GF(q)$. It is denoted $\mathrm{Rk}(\mathbf{c} \mid GF(q))$.

In the same way, given a code over $GF(q^m)$, the minimum rank distance of the code is the quantity

$$d = Min_{\mathbf{c} \in C \setminus \{0\}}(\mathrm{Rk}(\mathbf{c} \mid GF(q)))$$

Let C be a linear code with parameters (n, k), and minimum rank distance d over $GF(q^m)$. In rank metric the problem of bounded distance decoding of a code can be formulated as such

Decoding$(\mathbf{y}, C, t)$
Find, when it exists, $\mathbf{c} \in C$*, and* $\mathbf{e}$ *where* $Rk(\mathbf{e} \mid GF(q)) \leq t$ *such that* $\mathbf{y} = \mathbf{c} + \mathbf{e}$*,*

where $\mathbf{y}$ is the received vector over $GF(q^m)$, C is a code over $GF(q^m)$, and t is a positive integer. Provided t is less than or equal to the rank error-correcting capability of the code C, either there is no solution or the solution is unique.

Some general purpose decoding algorithms were constructed, for example in [2] but the best ones were designed by Ourivski and Johannson in [9]. Both are based on writing a set of quadratic equations satisfied by the error-vector, and linearizing a part of it by some extended search over a definite vector space. Provided one wants to correct t rank errors over $GF(q^m)/GF(q)$ in a code of length n, dimension k, their complexity is given by:

- *First strategy:* $\mathrm{O}((mt)^3 q^{(t-1)(k+1)})$ operations in $GF(q)$.
- *Second strategy:* $\mathrm{O}((k+t)q^{(t-1)(m-t)})$ operations in $GF(q)$.

It is highly exponential. Therefore, given a code C, we are not generally able to solve the **Decoding** problem for the code C, even for small parameters. This property enables to design Public-Key cryptosystems based on codes with theoretically a smaller public-key size than in Hamming metric [5].

In the seminal paper, Gabidulin presented a new family of codes defined by a vector $\mathbf{g} = (g_1, \ldots, g_n)$ of elements of $GF(q^m)$ linearly independent over $GF(q)$. A generating matrix of such a code $Gab_k(\mathbf{g})$ is the matrix G such that

$$G = \begin{pmatrix} g_1 & \cdots & g_n \\ \vdots & \ddots & \vdots \\ g_1^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix},$$

These codes are called Gabidulin codes and are denoted $Gab_k(\mathbf{g})$. They have minimum rank distance $d = n - k + 1$ and possess fast-polynomial time decoding algorithm. Namely, if we instantiate the problem **Decoding**$(\mathbf{y}, C, t)$ with a Gabidulin code of minimum distance d and with $t \leq \lfloor (d-1)/2 \rfloor$, there are fast polynomial time decoding algorithms solving the problem. They are similar to corresponding decoding algorithms for Reed-Solomon codes:

- *Extended Euclidian like* : $\approx t(m + 2n + t^2)$ multiplications in $GF(q^m)$, see [11,4];
- *Berlekamp–Massey like*: $\approx t(m + 2n + 6t + t^2/2)$ multiplications in $GF(q^m)$, see [10].

3 The Reconstruction of q–Polynomials

q–polynomials (also called *linearized polynomials*) are polynomials of the form

$$P(x) = \sum_{i=0}^{t} p_i x^{q^i}, \quad \forall i,\ p_i \in GF(q^m),\ p_t \neq 0.$$

the integer t is called q–degree of P and is denoted $deg_q(P)$.

Gabidulin codes play the same role in rank metric as Reed-Solomon codes in Hamming metric. Namely, they are evaluation codes of q–polynomials, as defined by Øre [7,8], on a set of n elements taken from $GF(q^m)$, linearly independent over the base field $GF(q)$. Therefore it is natural to link a so-called *Reconstruction Problem* for q–polynomials to the decoding problem in rank metric. Here is the statement of the problem as presented in [6].

$\mathbf{Reconstruction}(\mathbf{y} = (y_1, \dots, y_n), \mathbf{g} = (g_1, \dots, g_n), k, t)$
Find the set (V, f) where V is a non-zero q–polynomial of q-degree $\leq t$ and where f is a q–polynomial of q-degree $< k$, such that

$$V(y_i) = V[f(g_i)], \text{ for all } i = 1, \dots, n.$$

This problem can be related to the problem of bounded distance decoding Gabidulin codes, by the following theorem.

Theorem 1. *From any solution to* $\mathbf{Reconstruction}(\mathbf{y}, \mathbf{g}, k, t)$*, where the g_i's are linearly independent over $GF(q)$ one gets a solution to* $\mathbf{Decoding}(\mathbf{y}, Gab_k(\mathbf{g}), t)$ *in polynomial time.*

Proof. Let $\mathcal{L}$ be the set of solutions of $\mathbf{Reconstruction}(\mathbf{y}, \mathbf{g}, k, t)$. Let $(V_1, f_1) \in \mathcal{L}$. Then for all $i = 1 \dots, n$ we have $V_1(y_i) = V_1[f_1(g_i)]$. By linearity of V_1, we get $V_1(y_i - f_1(g_i)) = 0$, for all $i = 1 \dots, n$. This is equivalent to the fact that for all $i = 1 \dots, n$, the field element $e_i \overset{def}{=} y_i - f_1(g_i)$ belong to a vector space over $GF(q)$ of dimension at most the q-degree of V_1, that is t. Therefore, the vector $\mathbf{e} = (e_1, \dots, e_n)$ is of rank at most t and $(\mathbf{c}, \mathbf{e})$ where $\mathbf{c} = (f_1(g_1), \dots, f_1(g_n))$ is a solution of $\mathbf{Decoding}(\mathbf{y}, Gab_k(\mathbf{g}), t)$. All these transformation can clearly be computed in polynomial time. □

Therefore, designing algorithms for reconstructing q–polynomials will enable us to solve the decoding problem in rank metric.

4 Solving the Reconstruction Problem

Suppose we are given,

- A vector $\mathbf{y} = (y_1, \dots, y_n)$ of elements taken over the field $GF(q^m)$;
- A vector $\mathbf{g} = (g_1, \dots, g_n)$ of elements taken over the field $GF(q^m)$, that are linearly independent over $GF(q)$;
- Integers $k,\ t$;

To solve **Reconstruction**$(\mathbf{y}, \mathbf{g}, k, t)$, we need to find the q-polynomials V of q–degree less than or equal to t, and f of q–degree less than k such that

$$V(y_i) = V[f(g_i)], \text{ for all } i = 1, \ldots, n. \tag{1}$$

It is a quadratic system of n equations in $t+1+k$ variables. Basically we have no clue on how to solve this system. A way would be to compute the Gröbner basis of the system by adding the field equations, and then extract the finite number of solutions by computing the number of points of the obtained variety. However we have no precise complexity results on the difficulty the computation.

It is the reason why we consider the following system: Find (V, N), a pair of q-polynomials, such that

$$\begin{cases} V(y_i) = N(g_i), & \forall i = 1, \ldots, n \\ deg_q(V) \le t, \\ deg_q(N) \le k+t-1, \end{cases} \tag{2}$$

This system is a linear system whose unknowns are the $k+2t+1$ coefficients of N and V. The following proposition gives a relation between the sets of solutions of the two systems

Proposition 1. *Any solution (V, p) of (1) provides a solution $(V, N = V \circ p)$ to* (2).

Proof. Let (V_0, p_0) be a solution of (1), then the pair $(V_0, N_0 = V_0 \circ p_0)$ is a solution of (2).

Moreover, in some cases there is reciprocity.

Proposition 2. *If $t \le (n-k)/2$) and if there is at least a non-zero solution to 1), then the dimension of the vector space of solutions of (2) has dimension equal to 1, and any non zero solution to (2) provides a solution to (1).*

Proof. Suppose that the dimension of the vector space of solutions of (2) is 0. Then the unique solution to the system is $(0, 0)$. But from Proposition 1 it implies that the only solution to (1) is equally $(0, 0)$.

Now let us consider a non-zero solution (V_0, p_0) of 1) then any solution V, N of (2) satisfies the following system of equations:

$$V_0 \left[N(g_i) - V \circ p_0(g_i) \right] = 0, \forall i = 1, \ldots, n$$

the q–polynomial $V_0 \left[N - V \circ p_0 \right](x)$ has q–degree less than or equal to $k+2t-1$. Since $t \le (n-k)/2$, this implie that it has degree less than or equal to $n-1$. Therefore as q–polynomials, we have $V_0 \left[N - V \circ p_0 \right](x) = 0$, and since q–polynomials form an integral domain for composition, we get that $N = V \circ p_0$. Moreover, this gives easily that there is some $\alpha \in GF(q^m)$ such that $(V, N) = \alpha(V_0, V_0 \circ p_0)$. Hence the set of solutions to (1) has the form $(\alpha V_0, p_0)$.

5 New Decoding Algorithms

Suppose we receive a vector $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in Gab_k(\mathbf{g})$ and $\mathbf{e}$ has rank less than or equal to the error-correcting capability of the code. From Proposition 2 it follows that, we only need to find one solution of the linear system (2) to get the unique solution of $\mathbf{Reconstruction}(\mathbf{y}, \mathbf{g}, k, t)$. Once we get this solution we can decode easily by merely computing a Euclidian division of q-polynomials.

Namely the decoding algorithm can be described as such:

1. Find a two q-polynomials (V_0, N_0) which are solution of (2);
2. Compute the Euclidian division of N_0 by V_0 and set $f = N_0/V_0$. We have
$$y_i = f(g_i) + e_i,$$
for all $i = 1, \ldots, n$.

The rest of the section is devoted to the description two different algorithms solving system (2).

The second step of the algorithm is not considered here since it was already shown by Øre that the division could be computed in polynomial time. In [7], he designed an algorithmic way of computing the Euclidian division of q–polynomials.

The complexity of computing the Euclidian division between N_0 and V_0 is $(k-1)t$ multiplications in $GF(q^m)$.

5.1 A Natural Algorithm

Let $\mathcal{V} \overset{def}{=} (v_0, \ldots, v_t)^T$, where the v_i's are the coefficients of the q-polynomial V and $\mathcal{N} \overset{def}{=} (n_0, \ldots, n_{k+t-1})^T$ where the n_i are the coefficients of the q-polynomial N. Set

$$S = \left.\begin{pmatrix} g_1 & \cdots & g_1^{[k+t-1]} & y_1 & \cdots & y_1^{[t]} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ g_n & \cdots & g_n^{[k+t-1]} & y_n & \cdots & y_n^{[t]} \end{pmatrix}\right\} n$$

Solving (2) is equivalent to solving the system

$$S \times \begin{pmatrix} \mathcal{N} \\ \mathcal{V} \end{pmatrix} = 0. \tag{3}$$

In the unknowns $\mathcal{N}$ and $\mathcal{V}$. Therefore it costs roughly $(k+2t)^3$ operations over $GF(q)$. It is far too much to be efficiently implemented, compared to the already existing decoding algorithms.

By considering (3), it is clear that a part of the matrix S is independent of the received word, depending only on the parameters of the Gabidulin code.

Let us write

$$S = \left(\begin{array}{c|c} G_1 & Y_1 \\ \hline G_2 & Y_2 \end{array}\right),$$

where $G_1 = \left(g_i^{[j]}\right)_{i=1,j=0}^{k+t,k+t-1}$ is the upper left $(k+t)\times(k+t)$ matrix of S. Since, by definition, the g_i's are linearly independent, G_1 an invertible matrix. Therefore solving (3) is equivalent to solving

$$\begin{cases} \mathcal{N} = U \times (Y_1\mathcal{V}), \\ ((T \times Y_1) + Y_2)\mathcal{V} = 0, \end{cases} \tag{4}$$

where $U = -G_1^{-1}$ and $T = -G_2G_1^{-1}$ can be precomputed. The complexity of this algorithm is thus $(k+t)(k+t^2+2t)+t^3/2$ operations over $GF(q^m)$. Even this complexity is not satisfactory compared to the complexity of the existing algorithms, see section 2.

5.2 A Trickier Algorithm

We will now design another algorithm solving the polynomial reconstruction problem. Although less natural it is also more efficient. Our goal consists in finding q–polynomials $V(y)$ of q–degree less than or equal to t and $N(x)$ of q–degree less than $k+t$ satisfying system (2), *i.e.*

$$V(y_i) - N(g_i) = 0, \quad \forall i = 1, \ldots, n.$$

The idea is to construct two sequences of polynomials $(V_0^{(i)}(y), N_0^{(i)}(x))$ and $(V_1^{(i)}(y), N_1^{(i)}(x))$, satisfying for $i \le n$ the following property denoted by $\mathcal{P}(i)$

$$\forall k \le i, \begin{cases} V_0^{(i)}(y_k) - N_0^{(i)}(g_k) = 0, \\ V_1^{(i)}(y_k) - N_1^{(i)}(g_k) = 0, \end{cases}$$

If we manage to bound the degrees of the polynomials such that

$$\begin{cases} deg_q\left(V_0^{(n)}\right) \le t \\ deg_q\left(N_0^{(n)}\right) \le k-1+t \end{cases} \quad \text{or} \quad \begin{cases} deg_q\left(V_1^{(n)}\right) \le t \\ deg_q\left(N_1^{(n)}\right) \le k-1+t \end{cases}$$

then we have won.

Since the label i runs over n positions, if we increase the degrees of the polynomials at each step then we will not be able to satisfy the condition on the degrees. Therefore a way must be found to keep the degrees as low as possible.

Suppose that we have constructed a sequence of polynomials satisfying $\mathcal{P}(j)$, for all $j = 0, \ldots, i < n$. We show how to build polynomials satisfying $\mathcal{P}(i+1)$. First we evaluate the following quantities.

$$s_0^{(i)} \overset{def}{=} V_0^{(i)}(y_{i+1}) - N_0^{(i)}(g_{i+1}),$$
$$s_1^{(i)} \overset{def}{=} V_1^{(i)}(y_{i+1}) - N_1^{(i)}(g_{i+1}).$$

These quantities correspond to some defect in what we expect. Namely, if both of them is equal to zero, then $\mathcal{P}(i+1)$ is immediately satisfied.

There are basically two manners to build polynomials satisfying $\mathcal{P}(i+1)$.

- First and most simple is to evaluate

$$N_0^{(i+1)}(x) = N_0^{(i)}(x)^p - s_0^{(i)} N_0^{(i)}(x),$$
$$V_0^{(i+1)}(y) = V_0^{(i)}(y)^p - s_0^{(i)} V_0^{(i)}(y),$$

  This corresponds for q–polynomials to the interpolation of the *multivariate* polynomial $Q(x,y) \overset{def}{=} V(y) - N(x)$ on the point $[(y_{i+1}, g_{i+1}), 0]$. We check that for all $k = 1 \dots i+1$, we have $V_0^{(i+1)}(y_k) - N_0^{(i+1)}(x_k) = 0$. It is important to note that this method increases the q–degree of non-zero polynomials by 1.
- The second one corresponds to cross evaluation. We set

$$N_1^{(i+1)}(x) = s_0^{(i)} N_1^{(i)}(x) - s_1^{(i)} N_0^{(i)}(x),$$
$$V_1^{(i+1)}(y) = s_0^{(i)} V_1^{(i)}(y) - s_1^{(i)} V_0^{(i)}(y).$$

  This transformation implies that $deg_q(N_1^{(i+1)}) \leq Max(deg_q(N_1^{(i)}), deg_q(N_0^{(i)}))$, with equality if the degrees of $N_1^{(i)}$ and $N_0^{(i)})$ are different. Therefore this does not increase the degrees and one can check that for all $k = 1 \dots i+1$, $V_1^{(i+1)}(y_k) - N_1^{(i+1)}(x_k) = 0$.

This is the heart of the decoding algorithm we design. Basically there will be steps where we increase the degrees of the polynomials by maintaining the degrees of the others constant.

Description of the Algorithm. The algorithm is described in Table **??**. We chose not to build the sequences $(N_0^{(i)}, V_0^{(i)})$ and $(N_1^{(i)}, V_1^{(i)})$, but to modify the considered polynomials. Hence we can save space. This implies that at every step i both pairs of polynomials (N_0, V_0) and (N_1, V_1) satisfy the property $\mathcal{P}(i)$.

The algorithm consists of three steps:

- *Precomputation step*:
  - Compute $Int(g_1, \dots, g_k)$, where $Int(g_1, \dots, g_k)$ denotes the unique monic polynomial of q–degree k such that $(Int(g_1, \dots, g_k)(g_i) = 0$, for all $i = 1, \dots, k$.
  - Compute the list $\mathcal{P}_i,\ i = 1, \dots, k$ of the k Lagrange interpolation polynomials of q–degree $k-1$, that is

$$\forall\ i = 1, \dots, k, \begin{cases} \mathcal{P}_i(g_j) = 0, \ mboxifj \neq i, \\ \mathcal{P}_i(g_i) = 1. \end{cases}$$

    This set of q–polynomials form a basis of the vector space of q–polynomials of q–degree $k-1$.

  For computation, we can use algorithms described by Øre in his paper for example.
- *Initialisation step*:
  - Set $V_0 = 0$, and $N_0 = Int(g_1, \dots, g_k)$.
  - Set $V_1(y) = y$ and

$$N_1 = \sum_{i=1}^{k} y_i \mathcal{P}_i.$$

From the properties of the polynomials $\mathcal{P}_i$, the polynomial N_1 has q–degree $k-1$ and satisfies the relations

$$\forall\, i = 1, \ldots, k, \quad N_1(g_i) = y_i.$$

– *Alternate increasing degree step*: This is the most delicate part of the algorithm. Indeed this part consists of checking the degrees of the pairs of polynomials. We now exchange the roles of N_0 and N_1 and V_0 and V_1, so that we will always increase the degree of N_0 and V_0 by one at each step. If we set $s = \lfloor (i-k)/2 \rfloor$, after the ith step we have
  - $deg_q(N_0) = k + s$;
  - $deg_q(V_0) = s$ if $i-k$ is even and $deg_q(V_0) = s+1$ if $i-k$ is odd;
  - $deg_q(N_1) = k+s-1$ if $i-k$ is even and $deg_q(N_1) = k+s$ if $i-k$ is odd;
  - $deg_q(V_1) = s$.

  Therefore after the final step n the pair of polynomials (N_1, V_1) satisfy the condition for being a solution to system (2), since $deg_q(N_1) = k + \lfloor (n-k)/2 \rfloor - 1$ and $deg_q(V_1) = \lfloor (n-k)/2 \rfloor$.

5.3 Complexity Analysis of the Algorithm

The most complex operation is multiplying elements in finite fields compared to squaring and additioning.

– *Initialisation step*: the only polynomial that cannot be precomputed is N_1 consisting of a linear combination of interpolation polynomials. Hence, the complexity of computing N_1 is k^2 multiplications in $GF(q^m)$.

– *Alternate Incresing Degree step*: Let us evaluate the complexity of the algorithm at step $i \geq k+1$
  - Computation of s_0 and s_1: In any case, it is easy to check that either in the even of in the odd case, the computation it takes exactly $2i-1$ multiplications.
  - Computing $s_0N_1(x) - s_1N_0(x)$ and $s_0V_1(y) - s_1V_0(y)$ costs equally $2i-1$ multiplications.
  - Computing $N_0(x)^q - s_0N_0(x)$, and $V_0(x)^q - s_0V_0(x)$ costs i multiplications.

  Therefore, at every step $k+1 \leq i \leq n$, one has to compute $5i-2$ multiplications. Hence the total number of multiplications for this step is:

$$\sum_{i=k+1}^{n} 5i - 2 = \frac{5}{2}(n^2 - k^2) + \frac{n-k}{2} - 2,$$

  multiplications in $GF(q^m)$.

The overall complexity gives about $\frac{5}{2}n^2 - \frac{3}{2}k^2 + \frac{n-k}{2}$ multiplications.

Table 1. Algorithm for solving the linear system

```
INPUT: A Gabidulin code $Gab_k(\mathbf{g})$ of length $n$, and a vector $\mathbf{y} = (y_1, \dots, y_n)$ at rank
distance less than or equal to $t = \lfloor (d-1)/2 \rfloor$ from $Gab_k(\mathbf{g})$.
OUTPUT: A pair of polynomials $(N_1, V_1)$ satisfying system (2)

 1. Initialisation step:
      – $V_0(y) \leftarrow 0$ and $V_1(y) \leftarrow y$,
      – $N_0(x) \leftarrow Int(g_1, \dots, g_k)$ and $N_1(x) \leftarrow \sum_{i=1}^{k} y_i \mathcal{P}_i$.
 2. Alternate increasing degree step
    For $i \in \{k+1, \dots, n\}$ do
      – $s_0 \leftarrow V_0(y_i) - N_0(g_i)$ and $s_1 \leftarrow V_1(y_i) - N_1(g_i)$,
      – Exchange $N_0$ and $N_1$, $V_0$ and $V_1$, $s_0$ and $s_1$
      – Compute
         (a) $N_1(x) \leftarrow s_0 N_1(x) - s_1 N_0(x)$,
         (b) $V_1(y) \leftarrow s_0 V_1(y) - s_1 V_0(y)$,
         (c) $N_0(x) \leftarrow N_0(x)^q - s_0 N_0(x)$,
         (d) $V_0(y) \leftarrow V_0(y)^q - s_0 V_0(y)$.
 3. Return $(N_1, V_1)$.
```

6 Conclusion

We implemented both algorithms as well as the *extended Euclidian* algorithm in Magma language. It appears, that the first approach is not faster than the extended Euclidian, and has approximately the same complexity, a little less efficient nevertheless.

Computer simulations made in MAGMA show that our second algorithm with complexity $5/2n^2 - 3/2k^2$ is almost always faster than the *extended Euclidian.* The thing is that the complexity of the latter is roughly in $O(t^3 + 2nt)$. This implies that whenever, t is great, the complexity is cubic, whereas when t is small, then the dimension k can be high, Thus reducing the complexity of our algorithm.

References

1. E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent, Number 4,633,470, 1986.
2. F. Chabaud and J. Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '96*, volume 1163 of *LNCS*. Springer-Verlag, November 1996.
3. E. M. Gabidulin. Theory of codes with maximal rank distance. *Problems of Information Transmission*, 21:1–12, July 1985.
4. E. M. Gabidulin. A fast matrix decoding algorithm for rank-error correcting codes. In G. Cohen, S. Litsyn, A. Lobstein, and G. Zémor, editors, *Algebraic coding*, volume 573 of *LNCS*, pages 126–133. Springer-Verlag, 1991.

5. E .M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *LNCS*, pages 482–489. Springer-Verlag, 1991.
6. P. Loidreau. Sur la reconstruction des polynômes linéaires : un nouvel algorithme de décodage des codes de Gabidulin. *Comptes Rendus de l'Académie des Sciences: Série I*, 339(10):745–750, 2004.
7. Ø. Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35:559–584, 1933.
8. Ø. Ore. Contribution to the theory of finite fields. *Transactions of the American Mathematical Society*, 36:243–274, 1934.
9. A. Ourivski and T. Johannson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38(3):237–246, September 2002.
10. G. Richter and S. Plass. Error and erasure decoding of rank-codes with a modified Berlekamp-Massey algorithm. In *5th Int. ITG Conference on Source and Channel Coding (SCC 04)*, 2004.
11. R. M. Roth. Maximum-Rank array codes and their application to crisscross error correction. *IEEE Transactions on Information Theory*, 37(2):328–336, March 1991.
12. M. Sudan. Decoding Reed-Solomon codes beyond the error-correction diameter. In *Proceedings of the 35th Annual Allerton Conference on Communication, Control and Computing*, pages 215–224, 1997.