A new structural attack for GPT and variants

May 31, 2005

Abstract

In this paper we look at the Gabidulin version of the McEliece cryptosystem (GPT) and its variants. We propose a new polynomial time attack on the private key, which is applicable to all variants proposed so far, breaking some of them completely.

Keywords: public key cryptography, code based cryptography, rank distance codes, Gabidulin codes.

1 Introduction

The security of cryptosystems based on error correcting codes is connected to the hardness of the general decoding problem. In 1991 Gabidulin, Paramonov and Tretjakov proposed a variant of the McEliece scheme (GPT) [6] using *rank distance* codes instead of hamming distance codes. Smaller public-key sizes have been proposed for GPT than for the original McEliece cryptosystem, as general decoding algorithms are much slower for for the rank metric than for the hamming-metric.

Gibson developed two structural attacks for the GPT cryptosystem (see e.g. [4] and [7]) and proved the parameter sets proposed in [6] and [4] to be insecure. A drawback of Gibson's attacks is, that they have exponential runtime if the secret key was carefully chosen. Several variants of GPT have been proposed in order to avoid Gibson's attacks (see e.g. [1] and [10]).

In this paper we build a new structural attack on the GPT cryptosystem. This new attack runs in polynomial time and is applicable to all other GPT variants proposed so far.

The paper is structured as follows: First we give a short introduction to rank distance codes. Then we present the Niederreiter variant of the GPT cryptosystem [1] and show how to attack it. Finally we extend our attack to the GPT cryptosystem.

2 Rank distance codes

Rank distance codes were presented by Gabidulin in 1985 They are linear codes over the finite field $\mathbb{F}_{q^m}$ for q (power of a) prime and $m \in \mathbb{N}$. As their name suggests they use a special concept of distance.

Definition 2.1 Let $x = (x_1, \cdots, x_n) \in \mathbb{F}_{q^m}^n$ and $b_1, \cdots, b_m$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. We can write $x_i = \sum_{j=1}^m x_{ij} b_j$ for each $i = 1, \cdots, n$ with $x_{ij} \in \mathbb{F}_q$. The *rank norm* $\|\cdot\|_r$ is defined as follows:

$$\|x\|_r := \operatorname{rank}\left((x_{ij})_{1 \le i \le n,\ 1 \le j \le m}\right) .$$

The rank norm of a vector $x \in \mathbb{F}_{q^m}^n$ is uniquely determined (independent of the choice of basis) and induces a metric, called *rank distance*.

Definition 2.2 An (n, k)-code $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subvectorspace of the vector space $\mathbb{F}^n$. We call the code $\mathcal{C}$ an (n, k, d) rank distance code if $d = \min_{x,y \in \mathcal{C}} \|x - y\|_r$. The matrix $C \in \mathbb{F}^{k \times n}$ is a *generator matrix* for the (n, k) code $\mathcal{C}$ over $\mathbb{F}$, if the rows of C span $\mathcal{C}$ over $\mathbb{F}$. The matrix $H \in \mathbb{F}^{n \times (n-k)}$ is called *check matrix* for the code $\mathcal{C}$ if it is the right kernel of C. The code generated by $H^\top$ is called *dual code* of $\mathcal{C}$ and denoted by $\mathcal{C}^\perp$. We write $G^\perp = H^\top$.

In [9] Ourivski and Johansson presented an algorithm which solves the general decoding problem in $\mathcal{O}\left((m\frac{d-1}{2})^3 q^{(d-3)(k+1)/2}\right)$ operations over $\mathbb{F}_q$ for (n, k, d) rank distance codes over $\mathbb{F}_{q^m}$. A special class of rank distance codes are the *Gabidulin codes* for which an efficient decoding algorithm exists [4]. We will define these codes by their generator matrix.

Definition 2.3 Let $g \in \mathbb{F}_{q^m}^n$ be a vector s.t. the components g_i, $i = 1, \cdots, n$ are linearly independent over $\mathbb{F}_q$. This implies that $n \le m$. The (n, k, d) Gabidulin code $\mathcal{G}$ is the rank distance code with generator matrix

$$G = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^q & g_2^q & \cdots & g_n^q \\ \vdots & & \ddots & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix} \in \mathbb{F}_{q^m}^{k \times n}. \tag{1}$$

An (n, k) Gabidulin code $\mathcal{G}$ corrects $\left\lfloor \frac{n-k}{2} \right\rfloor$ errors and has a minimum distance of $d = n - k + 1$. The dual code of an (n, k) Gabidulin code is a $(n, n-k)$ Gabidulin code (see [4]). The vector g is said to be the *generator*

vector of the Gabidulin code $\mathcal{G}$. A decoding algorithm based on the "right Euclidian division algorithm" runs in $\mathcal{O}\left(d\log_2^2 d + dn\right)$ operations over $\mathbb{F}_{q^m}$ for (n,k,d) Gabidulin codes [4].

Throughout this paper we will use the following notation. We write $\mathcal{G} = \langle G \rangle$ if the linear (n,k)-code $\mathcal{G}$ over the field $\mathbb{F}$ has the generator matrix G. We will identify $x \in \mathbb{F}^n$ with $(x_1, \cdots, x_n), x_i \in \mathbb{F}$ for $i = 1, \cdots, n$. For any (ordered) subset $\{j_1, \cdots j_m\} = J \subseteq \{1, \cdots n\}$ we denote the vector $(x_{j_1}, \cdots, x_{j_m}) \in \mathbb{F}^m$ with x_J. Similary, we denote by $M_{.J}$ the submatrix of a $k \times n$ matrix M consisting of the columns corresponding to the indizes of J and $M_{J'.} = \left(\left(M^\top\right)_{.J'}\right)^\top$ for any (ordered) subset J' of $\{1, \cdots, k\}$. Block matrices will be given in brackets.

3 The Niederreiter variant of GPT

In this section, we briefly introduce the Niederreiter variant of the GPT cryptosystem presented in [1].

- **System Parameters:** $q, n \leq m$, k, l, where $l < k$.
- **Key Generation:** First generate the following matrices over $\mathbb{F}_{q^m}$:

  G: $k \times n$ generator matrix of an (n,k) Gabidulin code $\mathcal{G}$ over $\mathbb{F}_{q^m}$.
  S: $(n-k+l) \times (n-k+l)$ random non-singular matrix
  A: $l \times n$ random matrix with rank l over $\mathbb{F}_{q^m}$ and rank n over $\mathbb{F}_q$.

  Then, compute $e = \frac{n-k}{2}$ and the $k \times n$ matrix

  $$\left(H'\right)^\top = S \begin{bmatrix} G^\perp \\ A \end{bmatrix}.$$

  Further let $\mathcal{D}_\mathcal{G}$ be an efficient syndrome decoding algorithm for $\mathcal{G}$.
- **Public Key:** (H', e)
- **Private Key:** $(\mathcal{D}_\mathcal{G}, S, A)$ or (G, S, A) where G is of the form in (1).
- **Encryption:** To encode a plaintext $x \in \mathbb{F}_{q^m}^n$ of rank norm less then e, compute the ciphertext c as follows:

  $$c = xH' \ .$$

- **Decryption:** To decode a ciphertext c apply the syndrome decoding algorithm $\mathcal{D}_\mathcal{G}$ for $\mathcal{G}$ to the syndrome build from the first $n-k$ columns of $s = S^{-1}c^\top$.

Parameters			Size Public	WF general
m	k	l	Key (Bytes)	decoding
25	17	4	487	2^{71}
25	15	5	469	2^{82}
32	24	4	960	2^{93}

Figure 3.1: Parameter sets for the Niederreiter GPT

In all examples and figures we will choose $n = m$ and $q = 2$. Figure 3.1 shows public key sizes and approximate workfactors (operations over $\mathbb{F}_q$) for the fastest general decoding attack. Parameters were taken from [2]. Note that the matrix H' describes a $\mathbb{F}_{q^m}$-linear subcode of $\mathcal{G}$, i.e. the intersection of $\mathcal{G}$ with the left kernel of $A^\top$.

3.1 Attacking the Niederreiter variant of GPT

The Niederreiter variant of the GPT cryptosystem was first attacked by A. Ourivski in [8]. In this section we introduce a new attack, which we will need later on, to attack the original GPT cryptosystem. We show how to attack the Niederreiter version of GPT. We don't consider the dual code of the code described by the public key (as it was done e.g. in [2]), but the code itself.

Theorem 3.1 *Let H' be a public check matrix of an instance of the Niederreiter variant of GPT with parameters q, m, n, k and l, where the private matrix S was generated at random with no more conditions than being non-singular. If $k - l > 1$ and $n - k - 1 \geq \lceil l/(k - l - 1) \rceil$, then we may recover the private key corresponding to H' in $\mathcal{O}\left(n^3\right)$ operations over $\mathbb{F}_{q^m}$ with high probability.*

Proof. Let G be a generator matrix of an (n, k) Gabidulin code over $\mathbb{F}_{q^m}$ with $n \leq m$ and generating vector $g_1, \cdots, g_n$. Let $\bar{S} \in \mathbb{F}_{q^m}^{(k-l)\times k}$ be a matrix of full rank over $\mathbb{F}_{q^m}$. Then $((H')^\top = (G')^\perp, e = (n-k)/2)$ with $G' = \bar{S}G$ is a public key of an instance of the Niederreiter variant of the GPT cryptosystem.

Given the public key we choose $f \in \mathbb{N}$ with $n-k-1 \geq f \geq \lceil l/(k-l-1) \rceil$. (For the parameter sets proposed e.g. in [2], the choice of $f = 1$ will be sufficient.) Let $\bar{G}$ be the generator matrix of an $(n, k+f)$ Gabidulin code over

$\mathbb{F}_{q^m}$ with generator vector $g_1, \cdots, g_n$. Then we have

$$G' = \left[\ \bar{S}\ \middle|\ 0\ \middle|\ \cdots\ \middle|\ 0\ \right] \bar{G}.$$

For a matrix M let $M^{[j]}$ denote the result of rising every element of M to the power of j. To recover the Gabidulin code $\langle G \rangle$ we define the following matrix:

$$\dot{G}_f := \begin{bmatrix} G' \\ (G')^{[q]} \\ \vdots \\ (G')^{[q^f]} \end{bmatrix} \tag{2}$$

The matrix $\dot{G}_f$ is a generator matrix of $\langle \bar{G} \rangle$ with high probability, which can be seen easily by writing it as

$$\dot{G}_f = \begin{bmatrix} \bar{S} & 0 & \cdots & 0 \\ 0 & \bar{S}^{[q]} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \bar{S}^{[q^f]} \end{bmatrix} \bar{G}.$$

Employing e.g. the algorithm described in [4], we may recover the generator vector g and thus $G^\perp$ in $\mathcal{O}\left((k+f)^3\right)$ Operations over $\mathbb{F}_{q^m}$.

We still have to recover S and A from H' and $G^\perp$. In order to do so, we choose a set L_1 of l rows of H', s.t. $S_{L_1 L_2}$ with $L_2 = \{n-k+1, \cdots, n-k+l\}$ is invertible. This may be done easily by successively appending rows from H' to $G^\perp$, s.t. the rank increases with each row added. We may assume that $L_1 = L_2$ and thus

$$\left(H'\right)^\top = \left[\begin{array}{c|c} S' \begin{bmatrix} G^\perp \\ A \end{bmatrix} \\ \hline A \end{array}\right]$$

for some $S' \in \mathbb{F}_{q^m}^{(n-k)\times(n-k+l)}$. Knowing A, $G^\perp$ and H' we can solve this system of $(n-k)\cdot n$ equations and $(n-k)\cdot(n-k+l)$ variables, to recover S'. This may be done in $\mathcal{O}\left((n-k)^3\right)$ operations. ■

Figure 3.2 shows modified parameter sets for which the presented attack does not work. These parameters are not necessarily secure, anyway (compare [8]).

Parameters			Size Public	WF general
m	k	l	Key (Bytes)	decoding
32	24	20	448	2^{93}
64	52	47	2360	2^{288}

Figure 3.2: Modified parameter sets for the Niederreiter GPT

4 Attacking the GPT cryptosystem

The attack presented in the previous section may be extended to the GPT cryptosystem. This extension is applicable to all variants of GPT as well.

First we give a short description of a generalized version of the GPT cryptosystem (GGPT). The public generator matrix of an instance of the GPT cryptosystem may be described as

$$G' = S\left(\left[\begin{array}{c|c} X & 0 \end{array}\right] + G\right) T \in \mathbb{F}_{q^m}^{k \times n}$$

for a special $k \times t$ matrix X, S non-singular, G generator matrix of an (n, k) Gabidulin code and T an non-singular matrix over $\mathbb{F}_q$ [10]. In general the *distortion matrix* X is of rank t over $\mathbb{F}_q$ and of rank $s \leq t$ over $\mathbb{F}_{q^m}$. The matrix T is called *column scrambler*. To encrypt a plaintext $x \in \mathbb{F}_{q^m}^k$ we compute the ciphertext $c = xG' + z$, where z is a random error vector of rank norm $(n - k - t)/2$. At decryption, we apply the error correction algorithm for $G_{\cdot\{t+1,\cdots,n\}}$ to $\left(c \cdot T^{-1}\right)_{\{t+1,\cdots,n\}}$ to recover the plaintext. Example parameter sets are given in figure 4.3. In the original GPT cryptosystem from [6] the random error z has rank norm $(n-k)/2 - t$ and the error correction algorithm for $G \cdot T$ (which is an (n, k) Gabidulin code, too) is applied to c to recover the plaintext.

4.1 A first attack aproach

To attack the GPT cryptosystem for special parameter sets, it is sufficient to analyze the structure of the check matrix of the public generator matrix. The code generated by G' has a check matrix of the form

$$\left(H'\right)^\top = \left[\begin{array}{cc} 0 & \left(G_{\cdot\{t+1,\cdots,n\}}\right)^\perp \\ A_1 & A_2 \end{array}\right] \cdot \left(T^{-1}\right)^\top \tag{3}$$

for some matrices A_1 and A_2 of appropiate dimensions. An attacker could guess a set N_1 of $n-t$ rows of H' s.t. $\left(T^{-1}\right)_{N_1 N_2}$ with $N_2 = \{t+1, \cdots, n\}$ is

invertible. Because of the special structure of $\left(G_{\cdot\{t+1,\cdots,n\}}\right)^{\perp}$ we may assume without loss of generality that $\left(T^{-1}\right)_{N_1 N_2}$ is the identity matrix. The matrix $H'_{N_1\cdot}$ corresponds to an instance of the Niederreiter version of GPT as long as $k-t>1$, which is fulfilled for most parameter sets proposed (compare [7], [2] and [10]). If $k-t>1$ and $n-k-t-1>\lceil t/(k-t-1)\rceil$, we can try to recover $\left(G_{\cdot\{t+1,\cdots,n\}}\right)^{\perp}$ by applying the attack presented in the previous section. If the attack succeeds, it reveals the private key of the GPT cryptosystem as we are now able to determine an alternative column scrambler T.

For random instances of the GPT cryptosystem, the subcodes we get from $H'_{\cdot N_1}$ don't seem to be uniformly distributed subcodes of the underlying Gabidulin code. In our experiments we noticed that the success probability of our attack decreases, as s decreases. The attack of Ourivsky might have better performance for this instances of the Niederreiter GPT, but we did not make experiments on that. However, our results for the Niederreiter variant are not affected by this observation.

4.2 Generalization of the attack

The attack described above is limited to parameter sets, where $k-t>1$ and does not succed with satisfying probability if s is small. Using the approach described in the section 3.1 we want to augment k and the dimension of the dual code in a preprocessing step. Afterwards we apply our attack on the Niederreiter GPT to recover the private key.

Let $\bar{G}$ be the generator matrix of the $(n,k+f)$ Gabidulin code with generator vector build by the last $n-t$ entries of the generator of G and $f\leq n-k-t-1$. Let $\dot{G}_f$ the matrix build from G' as described in equation (2). Then $\dot{G}_f$ has a dual matrix of the form

$$\dot{G}_f^{\perp}=\begin{bmatrix} 0 & \left(\bar{G}\right)^{\perp} \\ B_1 & B_2 \end{bmatrix}\cdot\left(T^{-1}\right)^{\top}, \tag{4}$$

where B_1 is a $l\times t$ matrix with $l\leq t$. Again, a random set of $n-t$ rows of $\dot{G}_f^{\perp}$ are very likely to correspond to an instance of the Niederreiter GPT. Therefore we can try to apply the methods described in theorem 3.1 to recover an alternative private key. Note that if $l=0$, then $\dot{G}_f^{\perp}$ reveals enough information to recover an alternatrive column scrambler immediately.

Based on our experiments we make the following assumption for $s<k$:

Assumption 1 *The dual matrix of $\dot{G}_f$ is of the form given in equation (4), where the number of rows of B_1 is smaller than $t-fs$ with high probability.*

If this assumption is true, then l drops down to zero for most instances of the original GPT cryptosystem if we choose $f = n - k - t - 1$. In our experiments we did not find any counterexample for random instances. We are going to give more arguments on why we consider this assumption to be true in the following sections.

4.3 Analysis of the new attack

The presented attack only succeeds for all parameter sets of the original GPT cryptosystem with high probability, if assumption 1 is true. It is obvious, that if the attack succeeds in recovering an alternative private key, it runs in $\mathcal{O}\left(m^5\right)$ operations over $\mathbb{F}_q$.

To verify the crucial assumption, we have to estimate the rank of $\dot{G}_f$ for given $G' = S\left(\left[\begin{array}{c|c} X & 0 \end{array}\right] + G\right)T$ and f. Therefore we view the following matrices for $1 \leq i \leq f$:

$$\begin{aligned} &\left(\left(S^{[q^{i-1}]}\right)^{-1}(G')^{[q^{i-1}]}\right)_{\{2,\cdots,k\}} + \left(\left(S^{[q^i]}\right)^{-1}(G')^{[q^i]}\right)_{\{1,\cdots,k-1\}}, \\ &\quad = \left(\left(X^{[q^{i-1}]}\right)_{\{2,\cdots,k\}} + \left(X^{[q^i]}\right)_{\{1,\cdots,k-1\}}\right) \cdot T \end{aligned} \tag{5}$$

which are linear combinations of submatrices of $\dot{G}_f$. These matrices have rank $\geq \min(s, k-1)$ if the matrix

$$\dot{X}_i := \left[\begin{array}{c} X \\ X^{[q^1]} \\ \vdots \\ X^{[q^i]} \end{array}\right]$$

has rank $(f+1) \cdot s$, which is very probable if s or i is small. The rank of $\dot{G}_f$ thus should be larger than $R = \min\left(k + f + t, k + f + fs, k + fk\right)$ for random X. The latter can be seen, if we replace $(G')^{[q^i]}_{\{1,\cdots,k-1\}}$ in $\dot{G}_f$ by the matrix given in equation (5) for every $1 \leq i \leq f$. This corroborates assumption 1.

4.4 Experimental Results

Figure 4.3 shows absolute runtimes for this last version of our attack in comparison to the theoretical workfactors (operations over $\mathbb{F}_q$) of the previous attacks. For all parameter sets we chose $q = 2$, $m = n$ and $f = n - t - k - 1$.

Parameters				average runtime	WF best of	WF general
m	k	t	s	of our attack	Gibson's attacks	decoding
48	10	16	3	51 min	2^{139}	2^{134}
48	16	18	4	58 min	2^{200}	2^{124}
48	24	8	2	102 min	2^{122}	2^{198}

Figure 4.3: Attacking the GPT cryptosystem

Operations were performed on a 500Mhz Pentium III running Linux using an implementation in Java.

In our experiments we chose X as the product of a random $k \times s$ matrix S_X of rank $s < k$ over $\mathbb{F}_{q^m}$ and a random $s \times t$ matrix $\bar{X}$ (of rank s over $\mathbb{F}_{q^m}$ and rank t over $\mathbb{F}_q$). For such choices of X the matrix $\dot{G}_f$ almost always had rank $(k + f + (s + 1) \cdot \min(f, s) + s \cdot \max(0, f - s))$ or $k + f + t$. For special choices of S_X and random $\bar{X}$, we were able to create instances, where the rank of $\dot{G}_f$ reached the bound R. To reduce the rank of $\dot{G}_f$ even more, it seems, that we would have to choose $\bar{X}$ of a special form, too. The latter removes further degrees of freedom in choosing the private key and thus does not seem to be a good choice.

4.5 On Secure Instances of GGPT

We have seen, that instances of the GPT cryptosystem and its variants, where

$$t \leq s \cdot (n - t - k - 1)$$

holds, are insecure if assumption 1 holds. For the GGPT variant however, we may choose parameter sets, s.t. this equation does not hold. Even though, we might be able recover the private key if we can choose a f s.t. $k + f - l > 1$ and $n - k - t - f - 1 \geq \lceil l/(k + f - l - 1) \rceil$, where $l = t - fs$. If these conditions are fulfilled, a selection of $n - t$ columns of $\dot{G}_f^{\perp}$ corresponds to an instance of the Niederreiter GPT, and we may apply the methods described in theorem 3.1.

To get secure instances of the GGPT cryptosystem, one could try to choose parameters in a way, such that $l > f + k$ for every possible choice of f. The latter is e.g. the case, if

$$s \leq \frac{2t - n}{n - t - k}$$

A parameter set satisfying this condition would e.g. be $n = m = 64$, $k = 8$, $t = 40$ and $s = 1$ with a public key size of 3584 bytes. The attack in the given form is not applicable for such parameter sets. However, it seems very likely that the attack may be modified in such a way, that these parameter sets may be attacked, too.

5 Conclusion

The attacks proposed in this paper succeed with good probability. We conclude that the original GPT cryptosystem from [6] and the 2001 variant with column scrambler [3] may not be considered to be secure. Our attack can even be extended to the GPT cryptosystem using reducible rank codes (compare [5], [10]). After several attacks on the GPT cryptosystem and its variants, it seems to be difficult to name secure parameter sets for the GPT variant from [10], if there exist any. Even if we would consider the parameter set mentioned above to be secure, the GPT cryptosystem looses much of its advantages over the McEliece cryptosystem.

References

[1] T. Berger and P. Loidreau. Security of the niederreiter form of the GPT public-key cryptosystem. In *IEEE International Symposium on Information Theory, Lausanne, Suisse*. IEEE, July 2002.

[2] T.P. Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. to appear.

[3] E. M. Gabidulin and A. V. Ourivski. Column scrambler for the GPT cryptosystem. *Discrete Applied Mathematics*, 128(1):207–221, 2003.

[4] E.M. Gabidulin. On public-key cryptosystems based on linear codes. In *Proc of 4th IMA Conference on Cryptography and Coding 1993*, Codes and Ciphers. IMA Press, 1995.

[5] E.M. Gabidulin, A.V. Ourivski, B. Honary, and B. Ammar. Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12):3289–3293, 2003.

[6] E.M. Gabidulin, A.V. Paramonov, and O.V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Proc. Eurocrypt '91*, volume 547 of *LNCS*. Springer Verlag, 1991.

[7] K. Gibson. The security of the Gabidulin public key cryptosystem. In *Proc. of Eurocrypt'96*, volume 1070 of *LNCS*, pages 212–223. Springer Verlag, 1996.

[8] A. Ourivski. Recovering a parent code for subcodes of maximal rank distance codes. In *Proc. of WCC 03*, 2003.

[9] A.V. Ourivski and T. Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38, No. 3:237–246, 2002.

[10] R. Overbeck. Extending Gibson's attacks on the GPT cryptosystem. In *Proc. of WCC 2005*, 2005. to appear.