Eleventh International Workshop on Algebraic and Combinatorial Coding Theory
June 16-22, 2008, Pamporovo, Bulgaria *pp. 99-107*

Cryptanalysis of the McEliece cryptosystem over hyperelliptic codes

Cédric Faure cedric.faure@inria.fr
INRIA Rocquencourt

Lorenz Minder lorenz@eecs.Berkeley.edu
EECS Berkeley

Abstract. We present a practical expected usually quartic time algorithm to recover the structure of an algebraic geometry code defined over a hyperelliptic code of genus $g \leq 2$. Its main application is an attack of the McEliece cryptosystem based on algebraic geometry codes defined over curves of small genus. Our algorithm is a adaptation of the well-known Sidelnikov-Shestakov algorithm [6].

1 Introduction

In 1978, R. J. McEliece presented the first version of the cryptosystem which was to become the reference in public key cryptography based on coding theory [2]. The main version of McEliece's scheme uses Goppa codes. However, many other codes families have been studied to fit in McEliece's system.

The choice of a different code was often motivated by the goal to provide better security for a given key size. The most basic measure for security of that type of cryptosystem is the cost to decode the code with (a refined version of) information set decoding to the decoding bound with a given fixed key size (which is the amount of memory needed to store a generator matrix).

It is, however, important to be aware of the fact that this direct-decoding measure does not take into account the possibility that an attacker may attempt to recover the structure of the code instead of trying to break the system by attempting to decode an unstructured linear code. The possibility of doing so depends on the code that has been used in the construction. Efficient structural attacks have been developed for example against Reed Solomon codes by Sidelnikov and Shestakov [6], then against concatenated codes by Sendrier [5], and against Reed-Muller codes by Minder and Shokrollahi [4].

Since structural attacks tend to be very effective if applicable, a difficult task faced by the designer of McEliece-type cryptosystems is to chose a family of codes which has a good tradeoff between rate and correction capability (and is thus resistant against direct decoding attacks) while also being structurally secure.

When it comes to correction capability at a fixed rate, algebraic geometry codes are often the best known choice for a given set of parameters.[1]

The superiority of geometric codes makes them thus a seemingly excellent building block for McEliece type cryptosystems, and it is therefore important to the researchers in the field to know whether and to what extent these codes lead to secure cryptosystems.

In this paper, we show that curves of very low genus g (more precisely, $g \leq 2$) are a bad choice, and that they can be broken with very high probability in heuristic expected polynomial (usually quartic) time. Specifically, this breaks some of the Janwa-Moreno parameters [1]. A predecessor of this attack which worked for $g = 1$ was presented in [3] and was partly based on earlier, unpublished ideas by Bleichenbacher, Melnik and Shokrollahi. Since the case $g = 0$ (Reed Solomon codes) was taken care of by Sidelnikov and Shestakov [6], we restrict our attention in this paper to the case $g = 2$ and we present an algorithm which attacks the cryptosystem in time $O(n^4)$ binary operations, where n is the length of the ciphertext block.

For our attack to work, a few additional assumption on the code have to be made, such as the requirement that the blocklength n be reasonably close to maximal for the given curve. Since this covers the most interesting cases, it does not appear to be a severe restriction.

It is in principle possible to run our attack on hyperelliptic curves of genus larger than two, but in its current form only at a large cost in both running time and success probability. We have not investigated the question closely, and it may well be that already for hyperelliptic curves of genus 3, our attack is not all that interesting in its current form. Our preliminary opinion on the matter is that without substantial improvements, this kind of attack is not applicable to codes defined over sufficiently complicated curves.

In section 2, we recall mathematical concepts and definitions. In section 3, we present the McEliece cryptosystem in the setting of hyperelliptic codes, and in section 4, our attack will be exposed. We will then present our conclusions in section 5.

2 Definitions and notations

2.1 Notions of algebraic geometry

Let $\mathcal{X}$ be a hyperelliptic curve of genus $g = 2$ over $\mathbb{A}_2(\mathbb{F}_q)$, defined by the equation:

$$y^2 + G(x)y = F(x), \text{ with } \deg(F) = 2g + 1, \text{ and } \deg(G) \leq g.$$

[1]We ignore graph based codes here, because they are structurally weak.

A divisor Δ over $\mathcal{X}$ is a formal finite sum of points of $\mathcal{X}$ with positive and negative multiplicities:

$$\Delta = \sum_{P \in \mathcal{X}} n_P \langle P \rangle,\ n_P \in \mathbb{Z}.$$

The degree of a divisor is the sum of the multiplicities of the points in the divisor:

$$\deg(\Delta) = \sum_{P \in \mathcal{X}} n_P.$$

Any rational function f over $\mathcal{X}$ has an associated divisor $\mathrm{div}(f)$ which is obtained by adding every zero of f and subtracting every pole (counted with multiplicity in either case), i.e.,

$$\mathrm{div}(f) = \sum_{P \in \mathcal{X}} \mathrm{ord}_P(f) \langle P \rangle.$$

For every rational function, we have $\deg(\mathrm{div}(f)) = 0$, but the converse is not true in general. The Jacobian group of $\mathcal{X}$ is defined as the group:

$$\mathrm{Jac}(\mathcal{X}) = \text{Divisors of degree 0/divisors of rational functions.}$$

The Generalized Hasse-Weil theorem states that

$$\mathrm{Jac}(\mathcal{X}) \simeq \mathcal{G} = \frac{\mathbb{Z}}{d_1\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{d_{2g}\mathbb{Z}},\ \text{with } d_1 | \ldots | d_{2g},\ d_1 | q-1,$$

and that

$$(\sqrt{q}-1)^{2g} \le \|\sharp\mathcal{G}\| \le (\sqrt{q}+1)^{2g}.$$

2.2 Geometric codes

Let Δ be a divisor of degree $k+g-1$ over $\mathcal{X}$. We define the associated linear space $\mathcal{L}(\Delta)$:

$$\mathcal{L}(\Delta) = \{f \in \mathbb{F}_q(\mathcal{X}) | \mathrm{div}(f) + \Delta \ge 0\} \cup \{0\}$$

The Riemann-Roch theorem states that $\mathcal{L}(\Delta)$ is a vector space of dimension k if $k \ge g-1$. (We shall always assume $k \ge g-1$ in the sequel, the other case being of no interest for the problem at hand.)

Given a set $(P_1, \ldots, P_n)$ of distinct rational points on $\mathcal{X}$, we can now define the associated geometric code $\mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \ldots, P_n))$ as:

$$\mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \ldots, P_n)) = \{(f(P_1), \ldots, f(P_n)) | f \in \mathcal{L}(\Delta)\}$$

This is a linear code of length n, of dimension k, and minimal distance $d \ge n-k-g+1$. Furthermore, this code can be decoded in polynomial time up to its correction capability $t = \frac{n-k-g}{2}$. See, e.g., [7].

3 The McEliece cryptosystem on geometric codes

A McEliece type cryptosystem on geometric codes can be defined as follows: Fix blocklength n and dimension k. Select a random curve $\mathcal{X}$ of genus 2 having at least $n+1$ rational points, and randomly select distinct rational points $P_1, \dots, P_n$ on $\mathcal{X}$.

Let Δ be a divisor of degree $k+g-1$, and whose support is disjoint from the points P_i. This defines a code

$$\mathcal{C} := \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n)).$$

Now let G be a random generator matrix of $\mathcal{C}$, computed by multiplying a canonical generator matrix for this code on the left with an invertible $k \times k$-matrix with coefficients in $\mathbb{F}_q$.

This generator matrix G then serves as public key. The code parameters $\mathcal{X}, \Delta, (P_1, \dots, P_n)$ are the private key. A message $\mathbf{x} \in \mathbb{F}_q^k$ is encrypted by computing $\mathbf{y} := \mathbf{x}G + \mathbf{e}$, where $\mathbf{e}$ is a random weight $t = (n-k-g)/2$ error vector. The legitimate receiver who knows the secret parameters $\mathcal{X}$, Δ, $P_1, \dots, P_n$ can recover $\mathbf{x}$ by applying a decoding algorithm to $\mathcal{C}$, and thus computing $\mathbf{x}G$. Given $\mathbf{x}G$, the value of $\mathbf{x}$ can be recovered by solving a system of linear equations.

4 An attack against geometric codes of genus 2

Our goal is to recover a private key given the public key. In our setting this means the following: The attacker is given a generator matrix G of a hyperelliptic code $\mathcal{C}'$ of unknown parameters $\mathcal{X}', \Delta', (P'_1, \dots, P'_n)$. Inspecting G, he then finds a (typically different) set of parameters $\mathcal{X}, \Delta, (P_1, \dots, P_n)$, such that the code

$$\mathcal{C} := \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n))$$

is just a directional scaling of $\mathcal{C}'$, i.e., there are nonzero constants $c_1, \dots, c_n$ such that any codeword $(y_1, \dots, y_n) \in \mathcal{C}$ corresponds to a codeword $(c_1 y_1, \dots, c_n y_n) \in \mathcal{C}'$.

He then finds the scaling coefficients $(c_1, \dots, c_n)$, and this enables him to use the decoder for $\mathcal{C}$ to decode codewords for $\mathcal{C}'$, thus breaking the system.

4.1 Outline of the attack

Our algorithm works in four steps:

1. Recovering the group structure. In this stage we sample minimum weight codewords in order to collect linear equations on elements of the Jacobian. Given enough such relations, we can retrieve the finite group structure of the Jacobian $\mathcal{G}$.

2. Recovering the curve equation. We then use the Jacobian structure, along with a few additional carefully chosen minimum weight codewords, in order to get conditions on the coordinates of a few points of the curve.

   We then guess the coordinates of three points, and compute from this guess the coordinates of a larger set of points using the precomputed conditions. If the guess is correct, we can draw a hyperelliptic curve passing through our points. Otherwise we know that the guess is wrong, and we retry with another guess.

3. Recovering the coordinates of all the evaluation points. It is now possible, from the Jacobian structure, and the curve equation, to retrieve the coordinates of all points in the evaluation set.

4. Computing the scaling coefficients.

For all the steps to work, we will need additional assumptions. First, we assume that we have many evaluation points, i.e., that n is close to the number of rational points on $\mathcal{X}$.

Second, we will assume that $\gcd(k+g-1, |\mathcal{G}|) = 1$. Notice that we can force this latter condition to hold by working on a shortened version of the code, if necessary.

Third, we assume that the true minimum distance of the code is indeed $n-k-g+1$, and that many such codewords exist. Empirical evidence shows that this is virtually always true in our setting.

4.2 Preliminaries: code invariance

The parameters of a given geometric code are not unique : It is actually possible to generate the same code using a different evaluation set and a different divisor. This is very useful to the cryptanalyst, because it means that we can arbitrarily select some of the parameters we seek, and focus our search on the other ones.

In particular, we have $\mathrm{AGC}(\mathcal{X}', \Delta', (P'_1, \dots, P'_n)) = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n))$ if there exists a curve isomorphism from $\mathcal{X}'$ to $\mathcal{X}$ which maps P'_i to P_i and Δ' to Δ.

Let $u, v, a, b, c \in \mathbb{F}_q$. If $g = 2$, then the mapping

$$(x, y) \mapsto (u^2 x + v, u^{2g+1} y + ax^2 + bx + c)$$

is a curve isomorphism. If we arbitrarily select 2 points, and the Y-coordinate of a third, for example x_1, x_2, y_1, y_2, y_3 of P_1, P_2, P_3, then with probability 1/2 there exists such an isomorphism which maps P'_i to P_i.

So we can arbitrarily fix P_1, P_2, y_3, and still have

$$\mathrm{AGC}(\mathcal{X}', \Delta', (P'_1, \dots, P'_n)) = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \dots, P_n)).$$

for some $x_3, P_4, \ldots, P_n, \mathcal{X}, \Delta$ with probability 1/2.

Furthermore, because $\gcd(k+g-1, |\mathcal{G}|) = 1$, we can assume $\Delta = (k+g-1)\Delta_0$, where Δ_0 is a divisor of degree 1.

4.3 First step: recovering the Jacobian structure

We know that $\mathrm{Jac}(\mathcal{X}) \simeq \mathcal{G} = \frac{\mathbb{Z}}{d_1\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{d_{2g}\mathbb{Z}}$. In this step, we will recover the values of $d_1, \ldots, d_{2g}$, along with the images in $\mathcal{G}$ of some particular elements of the Jacobian by an unknown isomorphism φ.

We generate minimum weight codewords, i.e., we compute $\mathbf{x} \in \mathcal{C}$ such that $|\mathbf{x}| = n-k-g+1$. For general codes, finding minimum weight words is hard, but in our specific case we need only $O(n^2)$ operations in $\mathbb{F}_q$ on the average to find a single word with the desired property.

Let $\mathbf{x}$ be such a codeword, and write $f \in \mathcal{L}(\Delta)$ the associated rational function (i.e., the function such that $\mathbf{x}_i = f(P_i)$ for $1 \le i \le n$).

If $x_{i_1} = \cdots = x_{i_{k+g-1}} = 0$, then

$$f(P_{i_1}) = \cdots = f(P_{i_k+g-1}) = 0,$$

and so

$$\mathrm{div}(f) = \langle P_{i_1} \rangle + \cdots + \langle P_{i_{k+g-1}} \rangle - (k+g-1)\Delta_0. \qquad (1)$$

Notice that the minimality of $\mathbf{x}$ implies equality in (1) rather than just the greater-than relation that holds for any word and its associated function.

If we set $\tilde{z}_i = \varphi(\langle P_i \rangle - \Delta_0)$ in $\mathcal{G}$, we have

$$\sum_{j=0}^{k+g-1} \tilde{z}_{i_j} = 0.$$

With many (slightly more than n) equations of this form, we are able to recover the structure of the group $\mathcal{G}$, i.e., the values of $d_1, \ldots, d_{2g}$. This is true because a random system of overdetermined linear equations does not have any solution in $\mathbb{Z}/m\mathbb{Z}$ for arbitrary m; but our system has solutions in $\mathbb{Z}/d_i\mathbb{Z}$.

One technique to recover the d_i is thus to select several systems with only n equations, and compute the determinant of the associated matrix; it will always be a multiple of $d_1 \ldots d_{2g}$, and so we can find the d_i by taking the gcd of several such determinants. Only few determinants are needed in practice, but the computation of a determinant is usually $O(n^4)$ binary operations.

We can also solve the system to get the values of all the $\tilde{z}_i$. However, the map φ is still unknown at this point.

We now want to determine in $\mathcal{G}$ the value of $\delta_0 = \varphi(\Delta_0 - \langle \mathcal{O} \rangle)$. We use a statistical test to do this. Since most rational points are in the set $\{P_1, \ldots, P_n\}$, the probability that for a random index i, there is an index j, such that P_i and P_j are opposites of one another is $1 - \varepsilon$, where ε is upper bounded by twice the number of points not in $\{P_1, \ldots, P_n\}$ over the total number of rational points of $\mathcal{X}$. If P_i and P_j are indeed opposites, then $\langle P_i \rangle + \langle P_j \rangle = 2\langle \mathcal{O} \rangle$. It follows that $\tilde{z}_i + \tilde{z}_j = -2\delta_0$ in this case. For random indices i, j, the probability that $\tilde{z}_i + \tilde{z}_j = -2\delta_0$ is thus at least $(1-\varepsilon)/n$. At most $n/(1-\varepsilon)$ values can have that large a probability to be the equal to of $\tilde{z}_i + \tilde{z}_j$, so in the worst case we have $n/(1-\varepsilon)$ candidates for $-2\delta_0$, but we can expect that the set of sums behaves more randomly, and that we are able to extract the unique correct value of $-2\delta_0$ from the set of sums of all pairs. So we now know the value of δ_0, and we can compute all the values $z_i = \varphi(\langle P_i \rangle - \langle \mathcal{O} \rangle) = \tilde{z}_i + \delta_0$.

This test to recover the z_i usually runs in $O(n^2)$ multiplications over the base field.

4.4 Second step: Recovering the curve equation

We now generate two codewords $\mathbf{v}$ and $\mathbf{w}$ of weight $(n - k - g + 1)$, such that $\mathbf{v}$ and $\mathbf{w}$ have exactly $k + g - 3$ zero positions in common.

In order to do this, we first build a set $I \subset [1, n]$ of size $k + g - 3$ such that $\sum\limits_{i \in I} z_i = (k + g - 1)\delta_0$. We now select two couples of opposite points, i.e., $(i_1, i_2, j_1, j_2) \in [1, n]$ so that $z_{i_1} + z_{i_2} = z_{j_1} + z_{j_2} = 0$.

Then we know that there exists a codeword $\mathbf{v} \in C$ with zero positions on $I \cup \{i_1, i_2\}$. We can easily compute $\mathbf{v}$ from the generator matrix G. By the same method, we compute the codeword $\mathbf{w} \in C$ with zero positions on $I \cup \{j_1, j_2\}$.

Now that we have two such codewords $\mathbf{v}$ and $\mathbf{w}$, if we call f_1 and f_2, their respective (unknown) associated rational functions in $\mathcal{L}((k + g - 1)\Delta_0)$, then there exists $a, b, c, d \in \mathbb{F}_q$ so that $\frac{f_1}{f_2} = \frac{ax+b}{cx+d}$.

The functions f_1 and f_2 are unknown but, by definition of f_1 and f_2, for every i such that $w_i \neq 0$, we have $\frac{f_1}{f_2}(P_i) = \frac{v_i}{w_i} = \frac{ax_i+b}{cx_i+d}$.

All the v_i and w_i are known. So, if we know the coordinates (x_i, y_i) of three points (say, $P_{k_1}, P_{k_2}, P_{k_3}$), then we can first recover the constants a, b, c, d from the preceding equation on indexes k_1, k_2, k_3. We can then use those constants to recover the X-coordinates of many other P_i.

The Y-coordinate of P_i can be recovered by collinearity conditions: If, for example, we have $z_{k_1} + z_{k_2} + z_i + z_{i'} + z_{i''} = 0$ with a curve of genus $g = 2$, we can deduce that a straight line passes through P_{k_1}, P_{k_2}, P_i, $P_{i'}$ and $P_{i''}$. Then, if we know the coordinates of $P_{k_1}, P_{k_2}, P_{k_3}$, we can recover x_i from the preceding equation, and thereafter, y_i is deduced from the alignment of P_{k_1}, P_{k_2}, P_i.

So the indices k_1, k_2, k_3 must be chosen carefully. We will need $z_{k_1}, z_{k_2}, z_{k_3}$ to define three different 5-points collinearity equations, involving indexes which are non-zero positions of the word $\mathbf{w}$. We will also have to check that the set of 12 points involved in those collinearity equations generates the group $\mathcal{G}$.

Once these indices are chosen, we can guess and try the coordinates (x, y) of points $P_{k_1}, P_{k_2}, P_{k_3}$: We arbitrarily choose the values of their 6 coordinates. Then, with this set of values, we determine the constants a, b, c, d from our evaluation equation. Then, by the use of the evaluation equation and collinearity equations, we are able to determine the coordinates of 9 points P_i on the curve. We now try to build a hyperelliptic curve of genus 2 passing through our 12 points (the 3 we guessed, and the 9 we computed). If such a curve exists, then with great probability, our coordinates guess is correct and we proceed to the step 3 of our cryptanalysis. If such a curve does not exist, it means that our coordinates guess is wrong, and we try a new set of values $(x_{k_1}, y_{k_1}, x_{k_2}, y_{k_2}, x_{k_3}, y_{k_3})$.

Actually, we don't have many guesses to make in order to recover a valid curve. As we have seen precedently, if we arbitrarily choose $(x_{k_1}, y_{k_1}, x_{k_2}, y_{k_2}, y_{k_3})$ and try all the values for $x_{k_3} \in \mathbb{F}_q$, we have a probability 1/2 to obtain three points (P_1, P_2, P_3) such that there exists a curve isomorphism from $\mathcal{X}$ to $\mathcal{X}'$ which maps (P_1, P_2, P_3) to (P'_1, P'_2, P'_3). So, with q guesses, we have probability 1/2 to find a curve and a set of points so that $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, (k + g - 1)\Delta_0, (P_1, \ldots, P_n), (c_1, \ldots, c_n))$.

Since $q \approx n$, and processing one guess takes constant time, the total cost of the guessing step without preprocessing is $O(n)$ multiplications over the base field. The preprocessing, which consists of finding the words $\mathbf{v}$, $\mathbf{w}$ is $O(n^3)$ multiplications if a naive algorithm to find I is used.

4.5 Third and fourth step: recovering the remaining evaluation points and the distortion coefficients

We now know the equation of the hyperelliptic curve $\mathcal{X}$, along with the coordinates of a dozen points P_i on $\mathcal{X}$. We also know the values of all the $z_i = \varphi(\langle P_i \rangle)$ where φ is an unknown isomorphism.

The third step is then quite easy. For each P_i whose coordinates are still unknown, we write z_i as a sum of z_j corresponding to points whose coordinates are known. Computing the same sum with couples of points, we will find the coordinates of P_i. The value of the divisor Δ_0 will be computed the same way from the value of $\delta_0 = \varphi(\Delta_0 - \langle \mathcal{O} \rangle)$. The cost for computing the coordinates of one point is a constant, so the cost of this step is $O(n)$ multiplications over the base field.

When everything else is known, computing the distortion coefficients c_i is a simple linear algebra problem, which can be solved by a matrix inversion. The cost of this step is $O(n^3)$ multiplications, if we use a basic matrix inversion algortihm.

5 Conclusion

We have presented a polynomial time attack against a version of McEliece cryptosystem based on hyperelliptic codes of genus 2. As the first step of our algorithm has complexity $O(n^4)$ in the usual case, the complexity of the presented attack is $O(n^4)$ binary operations. Our attack is based on many probabilistic but reasonable assumptions, for example that the collected linear relations in step 1 behave like random relations modulo arbitrary integers.

Our attack is also restricted to the case where n is close to the number of rational points on the curve. We do not believe this to be a serious restriction. Ultimately, chosing small n is just one of many ways for a designer of a cryptosystem to trade efficiency for structural security, and quite possibly not the best one.

As it stands, the attack does not scale well with the genus. Indeed, for genus 3 the probability that the same attack works on a given instance is already quite low, even though certainly non-negligible. The fact, many of the steps of the attack work just fine also on these curves suggests that it is likely possible to devise almost-always working versions for these curves as well. It would be interesting to have a more thorough understanding of the fundamental limits of this kind of attack.

References

[1] H. Janwa, O. Moreno, McEliece public key cryptosystems using algebraic-geometric codes, *Des., Codes Crypt.* 8, 1996, 293-307.

[2] R. J. McEliece, A public key cryptosystem based on algebraic coding theory, DSN progress report 42-44, 1978, 114-116.

[3] L. Minder, Cryptography Based on Error Correcting codes Phd Thesis 3846, EPFL 2007,
`http://www.eecs.berkeley.edu/~lorenz/thesis.html`

[4] L. Minder, A. Shokrollahi, Cryptanalysis of the Sidelnikov cryptosystem, *Adv. Cryptology: Proc. EUROCRYPT 2007, LNCS.*

[5] N. Sendrier, On the structure of a randomly permuted concatenated code, *EUROCODE94*, October 1994.

[6] V. M. Sidelnikov, S. O. Shestakov, On insecurity of cryptosystems based on generalized Reed-Solomon codes, *Discr. Math. Appl.* 2, 1992, 439-444.

[7] S. G. Vlăduţ, On the decoding of algebraic-geometric codes over $\mathbb{F}_q$ for $q \geq 16$, *IEEE Trans. Inform. Theory* 36, 1990.