Practical Key Recovery Attacks On Two McEliece Variants

Valérie Gauthier Umaña and Gregor Leander

Department of Mathematics
Technical University of Denmark
Denmark
{v.g.umana, g.leander}@mat.dtu.dk

Abstract. The McEliece cryptosystem is a promising alternative to conventional public key encryption systems like RSA and ECC. In particular, it is supposed to resist even attackers equipped with quantum computers. Moreover, the encryption process requires only simple binary operations making it a good candidate for low cost devices like RFID tags. However, McEliece's original scheme has the drawback that the keys are very large. Two promising variants have been proposed to overcome this disadvantage. The first one is due to Berger et al. presented at AFRICACRYPT 2009 and the second is due to Barreto and Misoczki presented at SAC 2009. In this paper we first present a general attack framework and apply it to both schemes subsequently. Our framework allows us to recover the private key for most parameters proposed by the authors of both schemes within at most a few days on a single PC.

Keywords. public key cryptography, McEliece cryptosystem, coding theory, post-quantum cryptography

1 Introduction

Today, many strong, and standardized, public key encryption schemes are available. The most popular ones are based on either the hardness of factoring or of computing discrete logarithms in various groups. These systems provide an excellent and preferable choice in many applications, their security is well understood and efficient (and side-channel resistant) implementations are available.

However, there is still a strong need for alternative systems. There are at least two important reasons for this. First, RSA and most of the discrete log based cryptosystems would break down as soon as quantum computers of an appropriate size could be built (see [11]). Thus, it is desirable to have algorithms at hand that would (supposedly) resist even attackers equipped with quantum computers. The second reason is that most of the standard schemes are too costly to be implemented in very constrained devices like RFID tags or sensor networks. This issue becomes even more important when looking at the future IT landscape where it is anticipated that those tiny computer devices will be virtually everywhere [13].

One well-known alternative public encryption scheme that, to today's knowledge, would resist quantum computers is the McEliece crypto scheme [9]. It is based on the hardness of decoding random (looking) linear codes. Another advantage is that for encryption only elementary binary operations are needed and one might therefore hope that McEliece is suitable for constrained devices (see also [6] for recent results in this direction). However, the original McEliece scheme has a serious drawback, namely the public and the secret keys are orders of magnitude larger compared to RSA and ECC.

One very reasonable approach is therefore to modify McEliece's original scheme in such a way that it remains secure while the key size is reduced. A lot of papers already followed that line of research (see for example [10,7,2]), but so far no satisfying answer has been found. Two promising recent schemes in this direction are a McEliece variant based on quasi-cyclic alternant codes by

Berger et al. [4] and a variant based on quasi-dyadic matrices by Barreto and Misoczki [3]. As we will explain below both papers follow a very similar approach to find suitable codes. The reduction in the key size of both schemes is impressive and thus, those schemes seemed to be very promising alternatives when compared to RSA and ECC.

In this paper, however, we show that many of the parameter choices for both schemes can be broken. For some of them the secret key can be computed, given the public one, within less than a minute for the first variant and within a few hours for the second scheme. While there remain some parameter choices that we cannot attack efficiently, it seems that further analysis is needed before those schemes should be used.

Our attack is based on a general framework that makes use of (linear) redundancy in subfield subcodes of generalized Reed Solomon Codes. We anticipate therefore that any variant that reduces the key size by introducing redundancy in such (or related) codes might be affected by our attack as well.

We describe the two variants in Section 3 briefly, giving only the details relevant for our attack (for more details we refer the reader to the papers [4,3]). In Section 4 we outline the general framework of the attack and finally apply this framework to both schemes (cf. Section 5 and 6).

2 Notation

Let r, m be integers and $q = 2^r$. We denote by $\mathbb{F}_q$ the finite field with q elements and by $\mathbb{F}_{q^m}$ its extension of degree m. In most of the cases we will consider the case $m = 2$ and we stick to this until otherwise stated. For an element $x \in \mathbb{F}_{q^2}$ we denote its conjugate x^q by $\overline{x}$.

Given an $\mathbb{F}_q$ basis $1, \omega$ of $\mathbb{F}_{q^2}$ we denote by $\psi : \mathbb{F}_{q^2} \to \mathbb{F}_q^2$ the vector space isomorphism such that $\psi(x) = \psi(x_0 + \omega x_1) = \binom{x_1}{x_0}$. Note that, without loss of generality, we can choose θ such that $\psi(x) = \binom{\phi(x)}{\phi(\theta x)}$ where $\phi(x) = x + \overline{x}$ with $\overline{x} = x^q$. Note that we have the identity

$$\phi(\overline{x}) = \phi(x). \tag{1}$$

A fact that we will use at several instances later is that given $a = \phi(\alpha x)$ and $b = \phi(\beta x)$ for some $\alpha, \beta, x \in \mathbb{F}_{q^2}$ we can recover x as linear combination of a and b (as long as α, β form an $\mathbb{F}_q$ basis of $\mathbb{F}_{q^2}$). More precisely it holds that

$$x = \frac{\overline{\alpha}}{\beta\overline{\alpha} + \overline{\beta}\alpha} b + \frac{\overline{\beta}}{\beta\overline{\alpha} + \overline{\beta}\alpha} a \tag{2}$$

Adopting the notation from coding theory, all vectors in the papers are row vectors and vectors are right multiplied by matrices. The i.th component of a vector x is denote by $x^{(i)}$. Due to space limitations, we do not recall the basis concepts of coding theory on which McEliece and the two variants are based on. They are not needed for our attack anyway. Instead we refer the reader to [8] for more background on coding theory and in particular on subfield subcodes of Generalized Reed Solomon codes.

3 Two McEliece Variants

We are going to introduce briefly the two McEliece variants [4,3] that we analyzed. For this, we denote by x_i, c_i two sets of elements in $\mathbb{F}_{q^2}$ of size n. Furthermore let t be an integer. Both variants

have a secret key parity check matrix of the form:

$$\textbf{(secret key)} \quad H = \begin{pmatrix} \phi(c_0) & \phi(c_1) & \dots & \phi(c_{n-1}) \\ \phi(\theta c_0) & \phi(\theta c_1) & \dots & \phi(\theta c_{n-1}) \\ \vdots & \vdots & & \vdots \\ \phi(c_0 x_0^{t-1}) & \phi(c_1 x_1^{t-1}) & \dots & \phi(c_{n-1} x_{n-1}^{t-1}) \\ \phi(\theta c_0 x_0^{t-1}) & \phi(\theta c_1 x_1^{t-1}) & \dots & \phi(\theta c_{n-1} x_{n-1}^{t-1}) \end{pmatrix} = \begin{pmatrix} \mathrm{sk}_0 \\ \vdots \\ \mathrm{sk}_{2t-1} \end{pmatrix} \tag{3}$$

Thus, both schemes are based on subfield subcodes of Generalized Reed Solomon codes (see [8] for more background on those codes). Note that Goppa codes, the basic building block for the original McEliece encryption scheme are a particular kind of subfield subcodes of Generalized Reed Solomon codes. To simplify the notation later we denote by sk_i the i.th row of H.

The public key in both variants is

$$\textbf{(public key)} \quad P = SH, \tag{4}$$

where S is a secret invertible $2t \times 2t$ matrix. Actually, in both schemes P is defined to be the systematic form of H, which leads to a special choice of S. As we do not make use of this fact for the attacks one might as well consider S as a random invertible matrix.

In both cases, without loss of generality c_0 and x_0 can be supposed to be 1. In fact, given that the public key H is not uniquely defined, we can always include the corresponding divisions needed for this normalization into the matrix S.

The main difference between the two proposals is the choice of the constants c_i and the points x_i. In order to reduce the keysize, both of the public as well as of the secret key, those $2n$ values are not chosen independently, but in a highly structured way.

Both schemes use random block-shortening of very large private codes (exploiting the NP-completeness of distinguishing punctured codes [14]) and the subfield subcode construction (to resist the classical attack of Sidelnikov and Shestakov, see [12]). In [4,3] the authors analyze the security of their schemes and demonstrate that none of the known attack can be applied. They also prove that the decoding an arbitrary quasi-cyclic (reps. an arbitrary quasi-dyadic) code is NP-complete.

For the subfield subcode construction, both schemes allow in principle any subfield to be used. However the most interesting case in terms of key size and performance is the case when the subfield is of index 2 (i.e. $m = 2$) and we focus on this case only.

Both schemes use a block based description of the secret codes. They take b blocks of ℓ columns and t rows. The subfield subcode operation will transform each block into a $\ell \times 2t$ matrix and the secret parity check matrix H is the concatenation of the b blocks. Thus, one obtains a code of length ℓb.

Note that when we describe the variants, our notation will differ from the one in [4,3]. This is an inconvenience necessary in order to unify the description of our attack on both variants.

3.1 The Quasi-Cyclic Variant

Berger et al. propose [4] to use quasi-cyclic alternant codes over a small non-binary field. Let α be a primitive element of $\mathbb{F}_{q^m}$ and $\beta \in \mathbb{F}_{q^m}$ an element of order ℓ (those are public values). The secret

key consists of b different values y_j and a_j in $\mathbb{F}_{q^m}$ where b is small, i.e. $b \leq 15$ for the proposed parameters. The constants c_i and points x_i are then defined by

$$c_{\ell j+i} := \beta^{is} a_j \quad \text{and} \quad x_{\ell j+i} := \beta^i y_j \tag{5}$$

for all $0 \leq i \leq \ell - 1$ and $0 \leq j \leq b - 1$. Here $1 \leq s \leq \ell - 1$ is a secret value. Table 1 lists the parameters proposed in [4]. Note that in [4] cyclic shifts (modulo ℓ) of the columns are applied. This does not change the structure of the matrix (since β has order ℓ) and that is why we can omit this from our analysis.

Table 1. Parameters proposed in [4] and the running time/complexity of our attacks. The attacks were carried on a PC with an Intel Core2 Duo with 2.2 GHz and 3 GB memory running MAGMA version V2.15 − 12. Times are averaged over 100 runs.

	q	q^m	ℓ	t	b	Public key size (bits)	Assumed security	Complexity ($\log_2$) attack Section 5.1	Av. running time (sec) attack Section 5.2	Av. running time (sec) attack Appendix B
I			51	100	9	8160	80	74.9	–	–
II			51	100	10	9792	90	75.1	–	–
III	2^8	2^{16}	51	100	12	13056	100	75.3	–	–
IV			51	100	15	20400	120	75.6	–	–
V			75	112	6	6750	80	–	–	47
VI	2^{10}	2^{20}	93	126	6	8370	90	87.3	62	–
VII			93	108	8	14880	100	86.0	75	–

3.2 The Quasi-Dyadic Variant

Misoczki and Barreto propose [3] to use binary Goppa codes in dyadic form. They consider (quasi) dyadic Cauchy matrices as the parity check matrix for their code. However, it is well known that Cauchy matrices define generalized Reed Solomon codes [3] and thus, up to a multiplication by an invertible matrix which we consider to be incorporated in the secret matrix S, the scheme has a parity check matrix of the form (3).

Again, the only detail to be described here is how the constants c_i and points x_i are chosen. First we choose $\ell = t$ a power of two. Next, choose $v = [\mathbb{F}_{q^m} : \mathbb{F}_2] = mr$ elements in $\mathbb{F}_{q^m}$: $y_0, y_1, y_2, y_4, \cdots, y_{2^v}$. For each $j = \sum_{k=0}^{v} j_k 2^k$ such that $j_k \in \{0, 1\}$ (i.e. the binary representation of j) we define

$$y_j = \sum_{k=0}^{v} j_k y_{2^k} + (W_H(j) + 1) y_0 \tag{6}$$

for $0 \leq j \leq \#\mathbb{F}_{q^m} - 1$ and $W_H(j)$ is the Hamming weight of j. Moreover, choose b different elements k_i with $0 \leq i \leq \#\mathbb{F}_{q^m} - 1$, b different elements $a_i \in \mathbb{F}_{q^m}$ and define

$$x_{\ell i+j} := y_{k_i \oplus j} \quad \text{and} \quad c_{\ell i+j} := a_i \tag{7}$$

for all $0 \leq j \leq \ell - 1$ and $0 \leq i \leq b - 1$. This choice implies the following identity. For $j = \sum_{f=0}^{u-1} j_f 2^f$, where $u = \log_2(\ell)$ it holds that

$$x_{\ell i+j} = \sum_{f=0}^{u-1} j_f x_{\ell i+2^f} + (W_H(j) + 1) x_{li}. \tag{8}$$

Note that in [3] dyadic permutations are applied. However, this does not change the structure of the matrix and that is why we can omit this from our analysis.

Table 2 lists the parameters proposed in [3, Table 5].

Table 2. Sample parameters from [3] along with the complexity of our attack. Running time was measured on a PC with an Intel Core2 Duo with 2.2 GHz and 3 GB memory running MAGMA version V2.15 − 12.

q	q^m	ℓ	t	b	public key size	assumed security	complexity of the attack ($\log_2$)	estimated running time(h)
2^8	2^{16}	128	128	4	4096	80	43.7	36
		128	128	5	6144	112	43.8	41
		128	128	6	8192	128	44.0	47
		256	256	5	12288	192	44.8	107
		256	256	6	16384	256	44.9	125

4 General Framework of the Attack

The starting observation for our analysis and attacks is the following interpretation of the entries in the public key P.

Proposition 1. *Let H be the $2t \times n$ parity check matrix defined as in Equation (3). Multiplying H by a $2t \times 2t$ matrix S we obtain a $2t \times n$ matrix P of the form*

$$P = SH = \begin{pmatrix} \phi(c_0 g_0(x_0)) & \phi(c_1 g_0(x_1)) & \dots & \phi(c_{n-1} g_0(x_{n-1})) \\ \phi(c_0 g_1(x_0)) & \phi(c_1 g_1(x_1)) & \dots & \phi(c_{n-1} g_1(x_{n-1})) \\ \vdots & \vdots & & \vdots \\ \phi(c_0 g_{2t-1}(x_0)) & \phi(c_1 g_{2t-1}(x_1)) & \dots & \phi(c_{n-1} g_{2t-1}(x_{n-1})) \end{pmatrix}$$

where g_i are polynomials with coefficients in $\mathbb{F}_{q^2}$ of degree less than t. Moreover, if S is bijective the polynomials g_i form an $\mathbb{F}_q$ basis of all polynomials of degree at most $t-1$.

The proof of this proposition can be found in Appendix A.

This observation allows us to carry some of the spirit of the attack of Sidelnikov and Shestakov (see [12]) on McEliece variants based on Reed-Solomon codes over to the subfield subcode case. The basic idea is that multiplying the public key P by a vector results (roughly speaking) in the evaluation of a polynomial at the secret points x_i. More precisely the following holds.

Proposition 2. *Continuing the notation from above, multiplying the public parity check matrix P with a vector $\gamma \in \mathbb{F}_q^{2t}$ results in*

$$\gamma P = (\phi(c_0 g_\gamma(x_0)), \dots, \phi(c_{n-1} g_\gamma(x_{n-1}))) \tag{9}$$

where $g_\gamma(x) = \sum_{i=0}^{2t-1} \gamma_i g_i(x)$.

As the values γ, g_γ and γP are extensively used below we summarize their relation in Table 3.

Thus, if we would have the possibility to control the polynomial g_γ (even though we do not know the polynomials g_i) then γP reveals, hopefully, useful information on the secret key. While in general, controlling g_γ seems difficult, it becomes feasible in the case where the secret points x_i and the constants c_i are not chosen independently, but rather fulfil (linear) relations. The attack procedure can be split into three phases.

Table 3. The relation among the values γ, g_γ and γP. The polynomials g_i are defined in Proposition 1

γ	A vector in $\mathbb{F}_q^{2t}$
g_γ	The polynomial defined by $g_\gamma(x) = \sum_{i=0}^{2t-1} \gamma_i g_i(x)$.
γP	A vector in $\mathbb{F}_q^n$ whose entries are given by $\phi(c_i g_\gamma(x_i))$.

Isolate: The first step of the attack consists in choosing polynomials g_γ that we want to use in the attack. The main obstacle here is that we have to choose g_γ such that the redundancy allows us to efficiently recover the corresponding γ. As we will see later, it is usually not possible to isolate a single polynomial g_γ but rather to isolate a vector space of polynomials (or, equivalently, of vectors γ) of sufficiently small dimension.

Collect: After the choice of a set of polynomials and the recovery of the corresponding vectors γ, the next step of the attack consists in evaluating those polynomials at the secret points x_i. In the light of Proposition 2 this is simply done by multiplying the vectors γ with the public parity check matrix P.

Solve: Given the information collected in the second step of the attack, we now have to extract the secret key, i.e. the values x_i and c_i. This corresponds to solving a system of equations. Depending on the type of collected information this is done simply by solving linear equations, by first guessing parts of the key and then verifying by solving linear equations, or by solving non-linear equations with the help of Gröbner basis techniques. The advantage of the first two possibilities is that one can easily determine the running time in general while this is not true for the last one. However, the use of Gröbner basis techniques allows us to attack specific parameters very efficiently.

4.1 The Isolate Phase and the Collect Phase in Detail

The redundancy in the choice of the points x_i and the constants c_i will allow us to identify sets of polynomials or more precisely vector spaces of polynomials. In this section we elaborate a bit more on this on a general level. Assume that we are able to identify a subspace $\Gamma \subseteq \mathbb{F}_q^{2t}$ such that for each $\gamma \in \Gamma$ we know that g_γ is of the form

$$g_\gamma = \alpha_1 x^{d_1} + \alpha_2 x^{d_2} + \dots \alpha_r x^{d_r}$$

for some $\alpha_i \in \mathbb{F}_{q^2}$ and $d_i < t$. Equation (9) states that multiplying γ with the public key yields

$$\gamma P = \left(\phi(c_0 g_\gamma(x_0)), \dots, \phi(c_{n-1} g_\gamma(x_{n-1}))\right).$$

Using the assumed form of g_γ, and writing $\alpha_i = \alpha_{i,1} + \alpha_{i,2}\theta$ with $\alpha_{i,1}, \alpha_{i,2} \in \mathbb{F}_q$, we can rewrite $\phi(c g_\gamma(x))$ as

$$\begin{aligned}\phi(c g_\gamma(x)) &= \phi(c(\alpha_1 x^{d_1} + \alpha_1 x^{d_1} + \dots \alpha_r x^{d_r})) \\ &= \alpha_{1,1}\phi(cx^{d_1}) + \alpha_{1,2}\phi(\theta c x^{d_1}) + \dots + \alpha_{r,1}\phi(cx^{d_r}) + \alpha_{r,2}\phi(\theta c x^{d_r}).\end{aligned}$$

Recalling that we denote by sk_i the i.th row of the secret key (cf. Equation 3), we conclude that

$$\gamma P = \alpha_{1,1}\,\mathrm{sk}_{2d_1} + \alpha_{1,2}\,\mathrm{sk}_{2d_1+1} + \alpha_{2,1}\,\mathrm{sk}_{2d_2} + \alpha_{2,2}\,\mathrm{sk}_{2d_2+1} + \dots + \alpha_{r,2}\,\mathrm{sk}_{2d_r+1}\,.$$

Now, if the dimension of Γ is $2r$ this implies that there is a one to one correspondence between the elements $\gamma \in \Gamma$ and the coefficient vector $(\alpha_1, \ldots, \alpha_r)$. Stated differently, there exists an invertible $2r \times 2r$ matrix M such that for a basis $\gamma_1, \ldots, \gamma_{2r}$ of Γ we have

$$\begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_{2r} \end{pmatrix} P = M \begin{pmatrix} \mathrm{sk}_{2d_1} \\ \vdots \\ \mathrm{sk}_{2d_r+1} \end{pmatrix}, \tag{10}$$

where we now know all the values on the left side of the equation. This has to be compared to the initial problem (cf Equation 4) we are facing when trying to recover the secret key given the public one, where S is an invertible $2t \times 2t$ matrix. In this sense, the first step of the attack allows us to break the initial problem into (eventually much) smaller subproblems. Depending on the size of r (which will vary between 1 and $\log_2 t$ in the actual attacks) and the specific exponents d_i involved, this approach will allow us to efficiently reconstruct the secret key.

Note that we are actually not really interested in the matrix M, but rather in the values x_i and c_i. Therefore, a description of the result of the isolate and collect phase that is often more useful for actually solving for those unknowns is given by

$$M^{-1} \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_{2r} \end{pmatrix} P = \begin{pmatrix} \mathrm{sk}_{2d_1} \\ \vdots \\ \mathrm{sk}_{2d_r+1} \end{pmatrix}. \tag{11}$$

The advantage of this description is that the equations are considerably simpler (in particular linear in the entries of M^{-1}) as we will see when attacking specific parameters.

5 Applying the Framework to the Quasi-Cyclic Variant

In the following we show how the framework described above applies to the McEliece variant from [4] defined in Section 3.1. In particular we are going to make use of Equation (5). Recall that β is an element of order ℓ in $\mathbb{F}_{q^2}$. If ℓ is a divisor of $q-1$, such an element is in the subfield $\mathbb{F}_q$. This is the case for all the parameters in Table 1 except the parameter set V. We first focus on this case, the case that β is not in the subfield is considered in Appendix B. Section 5.1 describes an attack that works for parameters I-IV,VI and VII. Furthermore, for parameters VI and VII we describe attacks that allow us to recover the secret key within a few seconds in Section 5.2.

Note that in any case the secret value s (cf. Equation (5)) can be recovered very efficiently before applying the actual attacks, and we therefore assume it to be known from now on. However, due to space limitations and the fact that s is small anyway, we do not explain the details for recovering s.

5.1 The case $\beta \in \mathbb{F}_q$ (parameters I-IV,VI and VII)

In this part we describe an attack that works essentially whenever β is in the subfield. The attack has a complexity of roughly $q^6 \times (n_d b)(4n_d + b)^2(\log_2 q^2)^3$ (where $n_d = \lfloor \log_2(t-\ell) \rfloor$) which is lower than the best attacks known so far. Moreover, the attack is a key recovery attack, thus running the attack once allows an attacker to efficiently decrypt any ciphertext. However, these attacks are far from being practical (cf. Table 1 for actual values).

In the attack we apply the general framework twice. The first part will reduce the number of possible constants c_i to q^6 values. In the second part, for each of those possibilities, we try to find the points x_i by solving an over defined system of linear equations. This system will be solvable for the correct constants and in this case reveal the secret points x_i.

Recovering the Constants c_j

Isolate: We start by considering the simplest possible candidate for g_γ, namely $g_\gamma(x) = 1$. The task now is to compute the corresponding vector γ. Multiplying the desired vector γ with the public key P we expect (cf. Equation (9)) to obtain the following

$$\gamma P = (\phi(c_0 g_\gamma(x_0)), \ldots, \phi(c_{n-1} g_\gamma(x_{n-1}))) = (\phi(c_0), \phi(c_1), \ldots, \phi(c_{n-1})).$$

Now, taking Equation (5) into account, this becomes

$$\begin{aligned}\gamma P = \Big(&\phi(a_0), \phi(\beta^s a_0), \phi(\beta^{2s} a_0) \ldots, \phi(\beta^{(\ell-1)s} a_0),\\ &\phi(a_1), \phi(\beta^s a_1), \phi(\beta^{2s} a_1) \ldots, \phi(\beta^{(\ell-1)s} a_1),\\ &\vdots \quad \vdots\\ &\phi(a_{b-1}), \phi(\beta^s a_{b-1}), \phi(\beta^{2s} a_{b-1}) \ldots, \phi(\beta^{(\ell-1)s} a_{b-1})\Big).\end{aligned}$$

Since β is in the subfield we have $\phi(\beta x) = \beta\phi(x)$ for any $x \in \mathbb{F}_{q^2}$. Using this identity we see that γ corresponding to the constant polynomial g_γ fulfils

$$\gamma P = \phi(a_0)v_0 + \phi(a_1)v_1 + \cdots + \phi(a_{b-1})v_{b-1}$$

where

$$v_i = (\underbrace{0, \ldots, 0}_{i\ell}, 1, \beta^s, \beta^{2s}, \ldots, \beta^{(\ell-1)s}, \underbrace{0, \ldots 0}_{((b-1)-i)\ell}) \quad \text{for } 0 \le i \le b-1.$$

In other words, the γ we are looking for is such that γP is contained in the space U spanned by v_0 up to v_{b-1}, i.e. $\gamma P \in U = \langle v_0, \ldots, v_{b-1}\rangle$. Thus to compute candidates for γ we have to compute a basis for the space $\Gamma_0 = \{\gamma \mid \gamma P \in U\}$. We computed this space for many randomly generated public keys and observed the following.

Experimental Observation 1 *The dimension of the space Γ_0 is always 4.*

We do not prove this, but the next lemma explains why the dimension is at least 4.

Lemma 1. *Let γ be a vector such that $g_\gamma(x) = \alpha_0 + \alpha_1 x^\ell$. Then $\gamma \in \Gamma_0$.*

Proof. To show that γ is in Γ_0 we have to show that γP is a linear combination of the vectors v_i. To see this, it suffices to note that $g_\gamma(\beta x) = \alpha_0 + \alpha_1(\beta x)^\ell = \alpha_0 + \alpha_1 x^\ell = g_\gamma(x)$ as $\beta^\ell = 1$. As the points x_i fulfil Equation (5) we conclude $\gamma P = \phi(a_0 g_\gamma(y_0))v_0 + \phi(a_1 g_\gamma(y_1))v_1 + \cdots + \phi(a_{b-1} g_\gamma(y_{b-1}))v_{b-1}$. □

As, due to Observation 1, $\dim(\Gamma_0) = 4$ we conclude that

$$\{g_\gamma \mid \gamma \in \Gamma_0\} = \{\alpha_0 + \alpha_1 x^\ell \mid \alpha_0, \alpha_1 \in \mathbb{F}_{q^2}\}.$$

Collect Phase: Denote by $\gamma_1, \ldots, \gamma_4$ a basis of the four dimensional space Γ_0. Referring to Equation (11) we get

$$M^{-1} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix} P = \begin{pmatrix} \mathrm{sk}_0 \\ \mathrm{sk}_1 \\ \mathrm{sk}_{2\ell} \\ \mathrm{sk}_{2\ell+1} \end{pmatrix}. \tag{12}$$

for an (unknown) 4×4 matrix M^{-1} with coefficients in $\mathbb{F}_q$.

Solve Phase: We denote the entries of M^{-1} by (β_{ij}). The i.th component of the first two rows of Equation (12) can be rewritten as

$$\beta_{00}(\gamma_1 P)^{(i)} + \beta_{01}(\gamma_2 P)^{(i)} + \beta_{02}(\gamma_3 P)^{(i)} + \beta_{03}(\gamma_4 P)^{(i)} = \mathrm{sk}_0^{(i)} = \phi(c_i) = c_i + \overline{c_i}$$
$$\beta_{10}(\gamma_1 P)^{(i)} + \beta_{11}(\gamma_2 P)^{(i)} + \beta_{12}(\gamma_3 P)^{(i)} + \beta_{13}(\gamma_4 P)^{(i)} = \mathrm{sk}_1^{(i)} = \phi(\theta c_i) = \theta c_i + \overline{\theta c_i}.$$

Dividing the second equation by $\overline{\theta}$ and adding the two implies

$$\delta_0(\gamma_1 P)^{(i)} + \delta_1(\gamma_2 P)^{(i)} + \delta_2(\gamma_3 P)^{(i)} + \delta_3(\gamma_4 P)^{(i)} = \left(\frac{\theta}{\overline{\theta}} + 1\right) c_i, \tag{13}$$

where

$$\delta_i = \left(\beta_{0i} + \frac{\beta_{1i}}{\overline{\theta}}\right) \in \mathbb{F}_{q^2}.$$

Assume without loss of generality that $c_0 = 1$. Then, for each possible choice of δ_0, δ_1 and δ_2 we can compute δ_3 (using $c_0 = 1$) and subsequently candidates for all constants c_i. We conclude that there are $(q^2)^3$ possible choices for the constants c_i (and thus in particular for the b constants $a_0 = c_0, \ldots, a_{b-1} = c_{(b-1)\ell}$). We will have to repeat the following step for each of those choices.

Recovering Points x_i Given one out of the q^6 possible guesses for the constants c_i we now explain how to recover the secret values x_i by solving an (over defined) system of linear equations. Most of the procedure is very similar to what was done to (partially) recover the constants.

Isolate Here we make use of polynomials $g_\gamma = x^d$ for $d \leq t-1$. The case $g_\gamma = 1$ is thus a special case $d = 0$. Following the same computations as above, we see that for the vector γ corresponding to $g_\gamma = 1$ it holds that $\gamma P \in U_d$ where

$$U_d = \langle v_{(d)0}, \ldots, v_{(d)b-1} \rangle \tag{14}$$

and

$$v_{(d)i} = (\underbrace{0, \ldots, 0}_{i\ell}, 1, \beta^{s+d}, \beta^{2(s+d)}, \ldots, \beta^{(\ell-1)(s+d)}, \underbrace{0, \ldots 0}_{((b-1)-i)\ell}) \quad \text{for } 0 \leq i \leq b-1.$$

As before we define $\Gamma_d = \{\gamma \mid \gamma P \in U_d\}$, and, based on many randomly generated public keys we state the following.

Experimental Observation 2 *For $d \leq t - \ell$ the dimension of the space Γ_d is always* 4.

Similar as above, the next lemma, which can be proven similar as Lemma 1, explains why the dimension of Γ_d is at least 4.

Lemma 2. *Let γ be a vector such that $g_\gamma(x) = \alpha_0 x^d + \alpha_1 x^{d+\ell}$. Then $\gamma \in \Gamma_d$.*

As, due to Observation 2, $\dim(\Gamma_d) = 4$ we conclude that

$$\{g_\gamma \mid \gamma \in \Gamma_d\} = \{\alpha_0 x^d + \alpha_1 x^{d+\ell} \mid \alpha_0, \alpha_1 \in \mathbb{F}_{q^2}\}.$$

Collect Phase: Denote by $\gamma_{(d)1}, \ldots \gamma_{(d)4}$ a basis of the four dimensional space Γ_d. Referring to Equation (11) we get

$$M_d^{-1} \begin{pmatrix} \gamma_{(d)1} \\ \gamma_{(d)2} \\ \gamma_{(d)3} \\ \gamma_{(d)4} \end{pmatrix} P = \begin{pmatrix} \mathrm{sk}_{2d} \\ \mathrm{sk}_{2d+1} \\ \mathrm{sk}_{2(\ell+d)} \\ \mathrm{sk}_{2(\ell+d)+1} \end{pmatrix}$$

for an (unknown) 4×4 matrix M_d^{-1} with coefficients in $\mathbb{F}_q$ from which we learn (similar to Equation (13))

$$\big(\frac{\theta}{\bar{\theta}} + 1\big) c_i x_i^d = \delta_{(d)0}(\gamma_{(d)1} P)^{(i)} + \delta_{(d)1}(\gamma_{(d)2} P)^{(i)} + \delta_{(d)2}(\gamma_{(d)3} P)^{(i)} + \delta_{(d)3}(\gamma_{(d)4} P)^{(i)} \qquad (15)$$

for unknowns $\delta_{(d)i} \in \mathbb{F}_{q^2}$ (and unknowns x_i). How to solve such a system? Here, the freedom of choice in d allows us to choose $1 \leq d \leq t - \ell$ as a power of two. In this case, Equations (15) become *linear* in the bits of x_i when viewed as binary equations for a fixed guess for c_i. Let n_d be the number of possible choices for d, i.e. $n_d = \lfloor \log_2(t - \ell) \rfloor$. We get a linear system with $(\log_2 q^2)(4n_d + b)$ unknowns ($4n_d$ for the unknowns $\delta_{(d)i}$ and b unknowns for the points $x_{\ell j} = y_j$) and $(\log_2 q^2) n_d b$ equations ($\log_2 q^2$ equation for each d and each component $i = j\ell$). Thus whenever $b > 4$ and $n_d \geq 2$ (i.e. $t \geq 4$) this system is likely to be over defined and thus reveals the secret values x_i. We verified the behavior of the system and observed the following.

Experimental Observation 3 *Only for the right guess for the constants c_i the system is solvable. When we fix wlog $x_0 = 1$, for the right constants there is a unique solution for the values x_i.*

As there are q^6 possibilities for the constants and it takes roughly $(n_d b)(4n_d+b)^2(\log_2 q^2)^3$ binary operations to solve the system, the overall running time of this attack is $q^6 \times (n_d b)(4n_d+b)^2(\log_2 q^2)^3$. For the concrete parameters the attack complexity is summarized in Table 1.

5.2 Practical Attacks for parameter sets VI and VII

In this part we describe how, using Gröbner basis techniques, we can recover the secret key for the parameter sets VI and VII of Table 1 within a few seconds on a standard PC. The attack resembles in large parts the attack described above. The main difference in the solve phase is that we are not going to guess the constants to get linear equations for the points, but instead solve a non-linear system with the help of Gröbner basis techniques.

Isolate: Again, we make use of polynomials $g_\gamma = x^d$ but this time with the restriction $t-\ell \leq d < \ell$. To recover the corresponding vectors γ we make use of the space U_d defined by Equation (14). Now, with the given restriction on d it turns out that the situation, from an attacker's point of view, is nicer as for $\Gamma_d = \{\gamma \mid \gamma P \in U_d\}$, we obtain

Experimental Observation 4 *For $t-\ell \leq d < \ell$ the dimension of the space Γ_d is always 2.*

Thus, we isolated the polynomials $g(x) = \alpha_d x^d$ in this case. In other words

$$\{g_\gamma \mid \gamma \in \Gamma_d\} = \{\alpha x^d \mid \alpha \in \mathbb{F}_{q^2}\}.$$

The reason why we did not get the second term, i.e. $x^{d+\ell}$ in this case, is that the degree of g_γ is bounded by $t-1$ and $d+\ell$ exceeds this bound.

Collect Phase: Denote by $\gamma_{(d)1}, \gamma_{(d)2}$ a basis of the two dimensional space Γ_d. Referring to Equation (11) we get

$$M_d^{-1} \begin{pmatrix} \gamma_{(d)1} \\ \gamma_{(d)2} \end{pmatrix} P = \begin{pmatrix} \mathrm{sk}_{2d} \\ \mathrm{sk}_{2d+1} \end{pmatrix},$$

for an (unknown) 2×2 matrix M_d^{-1} with coefficients in $\mathbb{F}_q$.

Solve Phase: We denote the entries of M_d^{-1} by (β_{ij}). The i.th component of the first row can be rewritten as

$$\beta_{00}(\gamma_{(d)1}P)^{(i)} + \beta_{01}(\gamma_{(d)2}P)^{(i)} = c_i x_i^d + \overline{c_i x_i^d} \tag{16}$$

Again, we can assume $x_0 = c_0 = 1$. This (for $i = 0$) reveals $\beta_{00}(\gamma_{(d)1}P)^{(0)} + \beta_{01}(\gamma_{(d)2}P)^{(0)} = 0$ and thus $\beta_{01} = \frac{\beta_{00}(\gamma_{(d)1}P)^{(0)}}{(\gamma_{(d)2}P)^{(0)}}$. Substituting back into Equation (16) we get

$$\beta_{00}\left((\gamma_{(d)1}P)^{(i)} + \frac{(\gamma_{(d)1}P)^{(0)}}{(\gamma_{(d)2}P)^{(0)}}(\gamma_{(d)2}P)^{(i)}\right) = c_i x_i^d + \overline{c_i x_i^d}.$$

For parameter sets VI and VII we successfully solved this set of equations within seconds on a standard PC using MAGMA [5]. For parameters VI, d ranges from 33 to 92 and for parameters VII from 15 to 92. Thus in both cases we can expect to get a highly overdefined system. This allows us to treat $\overline{c_i}$ and $\overline{x_i^d}$ as independent variables, speeding up the task of computing the Gröbner basis by a large factor. An example code running the attack for parameter set VI can be found at [1]. The average running times are summarized in Table 1.

This attack does not immediately apply to parameters I to IV as here the range of d fulfilling $t-\ell \leq d < \ell$ is too small (namely $d \in \{49, 50\}$) which does not result in sufficiently many equations. However, we anticipate that using Gröbner basis techniques might speed up the attack for those parameters as well.

6 Applying the Framework to the Dyadic Variant

In this section we introduce, in a very similar way as we did in Section 5.1, how to apply the general framework of the attack to the McEliece variant introduced in [3] and described in Section 3.2. For $u = \log_2 t$ the attack to be described a complexity of roughly $q^2 \times (\log_2 q^2)^3(u^2+3u+b)^2u(u+b)$ binary operations, which for the parameters given in [3] means that we can recover the secret key within at most a few days with a standard PC (cf. Table 1 for actual values).

Recovering Constants c_j

Isolate phase: As before we consider $g_\gamma(x) = 1$ and we want to compute the corresponding vector γ. From Equation (9) we have that

$$\gamma P = (\phi(c_0 g_\gamma(x_0)), \ldots, \phi(c_{n-1} g_\gamma(x_{n-1}))) = (\phi(c_0), \phi(c_1), \ldots, \phi(c_{n-1})).$$

Now, taking Equation (7) into account, this becomes

$$\begin{aligned}\gamma P = (&\phi(a_0), \phi(a_0), \phi(a_0) \ldots, \phi(a_0),\\ &\phi(a_1), \phi(a_1), \phi(a_1) \ldots, \phi(a_1),\\ &\vdots \quad \vdots\\ &\phi(a_{b-1}), \phi(a_{b-1}), \phi(a_{b-1}) \ldots, \phi(a_{b-1}))\,.\end{aligned}$$

We see that γ corresponding to the constant polynomial g_γ fulfils

$$\gamma P = \phi(a_0)v_0 + \phi(a_1)v_1 + \cdots + \phi(a_{b-1})v_{b-1}$$

where

$$v_i = (\underbrace{0, \ldots, 0}_{i\ell}, 1, 1, 1, \ldots, 1, \underbrace{0, \ldots 0}_{((b-1)-i)\ell}) \quad \text{for } 0 \le i \le b-1.$$

Let U be the space spanned by v_0 up to v_{b-1}. The γ that we are looking for is such that

$$\gamma P \in U = \langle v_0, \ldots, v_{b-1} \rangle.$$

Thus in order to find γ we have to compute a basis for the space $\Gamma_0 = \{\gamma \mid \gamma P \in U\}$. We did this for many randomly generated public keys and observe the following.

Experimental Observation 5 *The dimension of the space Γ_0 is always* 2.

The next lemma shows, why the dimension is at least 2.

Lemma 3. *Let γ be a vector such that $g_\gamma(x) = \alpha_0$. Then $\gamma \in \Gamma_0$.*

Note that $\dim \Gamma_0 = 2$ is actually the best case we can hope for within our framework.

Collect Phase: Denote by γ_1, γ_2 a basis of the two dimensional space Γ_0. Referring to Equation (11) we get

$$M^{-1} \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} P = \begin{pmatrix} \mathrm{sk}_0 \\ \mathrm{sk}_1 \end{pmatrix} \tag{17}$$

for an (unknown) 2×2 matrix M^{-1} with coefficients in $\mathbb{F}_q$.

Solve Phase: We denote the entries of M^{-1} by (β_{ij}). We get

$$\begin{pmatrix} \beta_{00}\, \beta_{01} \\ \beta_{10}\, \beta_{11} \end{pmatrix} \begin{pmatrix} \gamma 1 \\ \gamma 2 \end{pmatrix} P = \begin{pmatrix} \phi(c_0), & \phi(c_1), & \cdots, & \phi(c_{b-1}) \\ \phi(\theta c_0), & \phi(\theta c_1), & \cdots, & \phi(\theta c_{b-1}) \end{pmatrix}.$$

Assuming wlog that $c_0 = 1$, we can compute β_{01} as a function of β_{00} and β_{11} as a function of β_{10}. Then guessing β_{00} and β_{10} allows us to recover all the constants. We conclude that there are q^2 possible choices for the b constants $a_0, \ldots, a_{b-1}$. We will have to repeat the following step for each of those choices.

Recovering Points x_i Assuming that we know the constants c_i we explain how to recover the secret values x_i by solving an (over-defined) system of linear equations. If the set of constants that we have chosen in the previous step is not the correct one, the system will not be solvable.

Isolate: We start by considering $g_\gamma(x) = x$, and multiply the desired vector γ with the public key P. We expect (cf. Equation (9)) to obtain the following:

$$\gamma P = (\phi(c_0 g_\gamma(x_0)), \ldots, \phi(c_{n-1} g_\gamma(x_{n-1})))$$

then

$$\begin{array}{rccc} \gamma P = & (\phi(a_0 x_0), & \phi(a_0 x_1), & \ldots, \quad \phi(a_0 x_{\ell-1}), \\ & \phi(a_1 x_\ell), & \phi(a_1 x_{\ell+1}), & \ldots, \quad \phi(a_1 x_{2\ell-1}), \\ & \vdots & \vdots & \vdots \\ & \multicolumn{3}{c}{\phi(a_{b-1} x_{(b-1)\ell}), \phi(a_{b-1} x_{(b-1)\ell+1}), \ldots, \phi(a_{b-1} x_{b\ell-1})).} \end{array} \tag{18}$$

Recalling Equation (8) we see that the vector γ we are looking for fulfils

$$(\gamma P)^{(\ell i + j)} = \sum_{f=0}^{u-1} j_f (\gamma P)^{(\ell i + 2^f)} + (1 + W_H(j))(\gamma P)^{(\ell i)} \quad \forall\ 0 \le i < b, 0 \le j < \ell \tag{19}$$

where $j = \sum_{f=0}^{u-1} j_f 2^f$ is the binary representation of j. Denoting $\Gamma_1 = \{\gamma \in \mathbb{F}_q^{2t} \mid \gamma \text{ fulfils (19)}\}$ we got the following observation by randomly generating many keys.

Experimental Observation 6 *The dimension of the space Γ_1 is always $u + 1$.*

Clearly, the dimension is at least $u + 1$ as we are actually only checking if g_γ is $\mathbb{F}_2$ affine and therefore if γ is such that $g_\gamma(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_u x^{2^{u-1}}$ then $\gamma \in \Gamma_1$.

Collect Phase: A straight-forward application of Equation (10) would lead to a linear system that becomes only over-defined for a large number of blocks. Thus, in order to avoid this we modify the collect phase as follows. Let $\gamma \in \Gamma_1$ be given. We have

$$\begin{aligned} \gamma P = (&\phi(a_0 g_\gamma(x_0)), \phi(a_0 g_\gamma(x_1)), \ldots, \phi(a_0 g_\gamma(x_{l-1})), \\ &\phi(a_1 g_\gamma(x_\ell)), \phi(a_1 g_\gamma(x_{\ell+1})), \ldots, \phi(a_1 g_\gamma(x_{2\ell-1})), \ldots) \end{aligned}$$

where g_γ is an $\mathbb{F}_2$ affine polynomial. Making use of the identity

$$x_0 + x_i = x_\ell + x_{\ell+i} \quad \forall\ 0 \le i < \ell$$

allows us to compute $\mu_\gamma^{(i)} = \phi(a_0(g_\gamma(x_0 + x_i) + g(0)))$ and $\nu_\gamma^{(i)} = \phi(a_1(g_\gamma(x_0 + x_i) + g(0)))$. As we assume we know the constants a_0 and a_1, given $\mu_\gamma^{(i)}$ and $\nu_\gamma^{(i)}$ we can recover (cf. Equation (2)) $z_\gamma^{(i)} = g_\gamma(x_0 + x_i) + g(0)$ (as long as (a_0, a_1) is an $\mathbb{F}_q$ basis of $\mathbb{F}_{q^2}$). Next, by solving a system of linear equations, we compute a γ' such that

$$z_{\gamma'}^{(i)} = \theta z_\gamma^{(i)}.$$

It turns out that the corresponding polynomial $g_{\gamma'}$ is unique up to adding constants, i.e. $g_{\gamma'} = \theta g_\gamma + c$. Summarizing our findings so far we get

$$\gamma P = (\phi(a_0 g_\gamma(x_0)), \phi(a_0 g_\gamma(x_1)), \ldots, \phi(a_{b-1} g_\gamma(x_{n-1})))$$
$$\gamma' P = (\phi(\theta a_0 g_\gamma(x_0) + a_0 c), \phi(\theta a_0 g_\gamma(x_1) + a_0 c), \ldots, \phi(\theta a_{b-1} g_\gamma(x_{n-1}) + a_{b-1} c)).$$

This, again using Equation (2), allows us to compute

$$\delta = (a_0 g_\gamma(x_0), a_0 g_\gamma(x_1), \ldots, a_{b-1} g_\gamma(x_{n-1})) + (a_0 c', a_0 c', \ldots, a_{b-1} c') + (\overline{a_0} c'', \overline{a_0} c'', \ldots, \overline{a_{b-1}} c'')$$

for (unknown) constants c', c''. Repeating this procedure for different elements $\gamma \in \Gamma_1$ will eventually result in $\delta_1, \ldots, \delta_{u+2}$ that span a space of dimension $u+2$. The data we collected can thus be written as

$$\begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{u+2} \end{pmatrix} = M \begin{pmatrix} (a_0, a_0, \ldots, a_{b-1}) \\ (\overline{a_0}, \overline{a_0}, \ldots, \overline{a_{b-1}}) \\ (a_0 x_0, a_0 x_1, \ldots, a_{b-1} x_{n-1}) \\ \vdots \quad \vdots \\ (a_0 x_0^{2^{u-1}}, a_0 x_1^{2^{u-1}}, \ldots, a_{b-1} x_{n-1}^{2^{u-1}}) \end{pmatrix} \tag{20}$$

for an invertible $(u+2) \times (u+2)$ matrix M.

Solve Phase: Multiplying Equation (20) by M^{-1} yields equations that, when viewed as binary equations, are linear in the entries of M^{-1} and the values x_i (as we assume the a_i to be known). The first two rows of M are determined by the (known) values of the constants a_i. Thus we are left with $N_u = \log_2(q^2)(u(u+2) + (u+b))$ unknowns, i.e. the remaining $u(u+2)$ entries of M^{-1} and the $u+b$ points

$$x_0, x_1, x_2, x_4, \ldots, x_{2^{u-1}}, x_\ell, x_{2\ell}, x_{3\ell}, \ldots x_{(b-1)\ell}$$

(all other points are given as linear combinations of those). The number of equations is $N_e = \log_2(q^2)(u+b) \times u$. In particular, whenever $b \geq 4$ and $u \geq 4$, i.e. $t \geq 2^4$, we get more equations than unknowns and can hope for a unique solution. We implemented the attack and observed the following.

Experimental Observation 7 *Only for the right guess for the constants c_i the system is solvable. In this case the constants x_0 and x_1 could be chosen as arbitrary non-zero elements in $\mathbb{F}_{q^2}$.*

As there are q^2 possibilities for the constants and it takes roughly $(N_e N_u^2)$ binary operations to solve the system, the overall running time of this attack is $q^2 \times (\log_2 q^2)^3 (u^2 + 3u + b)^2 u(u+b)$ binary operations. In Table 2 we computed the complexity of the attack for the sample parameters given in [3, Table 5]. A implementation of the attack in MAGMA can be obtained from [1]

Acknowledgement

We like to thank the authors of [4] for providing us a sample implementation of their cipher. We also like to thank Søren Thomsen and Tom Høholdt for fruitful discussions.

References

1. anonymous. Source Code for attacking two McEliece Variants. anonymous, October 2009.
2. M. Baldi and F. Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC Codes. *IEEE International Symposium on Information Theory*, pages 2591–2595, 2007.
3. Paulo S. L. M. Barreto and Rafael Misoczki. Compact McEliece Keys from Goppa Codes. In *Proceedings of the 16th International Workshop on Selected Areas in Cryptography, SAC 2009*, volume 5867 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.
4. Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. Reducing Key Length of the McEliece Cryptosystem. In Bart Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009.
5. Wieb Bosma, John J. Cannon, and Catherine Playoust. The Magma Algebra System I: The User Language. *J. Symb. Comput.*, 24(3/4):235–265, 1997.
6. T. Einsenbarth, T. Güneysu, S. Heyse, and C. Paar. MicroEliece: McEliece for Embedded Devices. In *Proceedings of CHES 2009*, LNCS. Springer, 2009. to appear.
7. P. Gaborit. Shorter keys for code based cryptography. International Workshop on Coding and Cryptography, Bergen, Norway, 2005.
8. F. J. MacWilliams and N. J. Sloane. *The theory of error-correcting codes*. North Holland, Amsterdam, 1977.
9. Robert .J. McEliece. A public key cryptosystem based on alegbraic coding theory. *DSN progress report*, 42-44:114–116, 1978.
10. C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. *IEEE International Symposium on Information Theory*, page 215, 2000.
11. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *IEEE Symposium on Foundations of Computer Science*, pages 124–134, 1994.
12. V. M. Sidelnikov and S. O. Shestakov. On cryptosystem based on gerenalized reed-solomon codes. *Discrete Mathematics*, 4(3):57–63, 1992.
13. Mark Weiser. The computer for the 21st century. *Scientific American*, Special Issue on Communications, Computers, and Networks September, September 1991.
14. C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. *IEEE International Symposium on Information Theory*, pages 1733–1737, 2006.

A Proof of Proposition 1

Proof. For the proof it is enough to consider the effect of multiplying a vector $s \in \mathbb{F}_q^t$ by H. For convenience we label the coordinates of s as

$$s = (\alpha_0, \beta_0, \alpha_1, \beta_1, \ldots, \alpha_{t-1}, \beta_{t-1})$$

We compute

$$\begin{aligned}
sH &= s \begin{pmatrix} \phi(\theta c_0) & \ldots & \phi(\theta c_{n-1}) \\ \phi(c_0) & \ldots & \phi(c_{n-1}) \\ \vdots & & \vdots \\ \phi(\theta c_0 x_0^{t-1}) & \ldots & \phi(\theta c_{n-1} x_{n-1}^{t-1}) \\ \phi(c_0 x_0^{t-1}) & \ldots & \phi(c_{n-1} x_{n-1}^{t-1}) \end{pmatrix} \\
&= \left(\sum_{i=0}^{t-1} \alpha_i \phi(\theta c_0 x_0^i) + \sum_{i=0}^{t-1} \beta_i \phi(c_0 x_0^i), \ldots, \sum_{i=0}^{t-1} \alpha_i \phi(\theta c_{n-1} x_{n-1}^i) + \sum_{i=0}^{t-1} \beta_i \phi(c_{n-1} x_{n-1}^i) \right) \\
&= \left(\phi(c_0 \sum_{i=0}^{t-1} (\theta \alpha_i + \beta_i) x_0^i), \ldots, \phi(c_{n-1} \sum_{i=0}^{t-1} (\theta \alpha_i + \beta_i) x_{n-1}^i) \right) \\
&= (\phi(c_0 g(x_0)), \ldots, \phi(c_{n-1} g(x_{n-1})))
\end{aligned}$$

where $g(x) = \sum_{i=0}^{t-1} (\theta \alpha_i + \beta_i) x^i$. □

B A practical attack for parameter set V

Recall that β is an element of order ℓ in $\mathbb{F}_{q^2}$. Attacks for the case that β is actually in the subfield $\mathbb{F}_q$ are discussed in Section 5. In the case that β is not in the subfield things are a little different and we focus on this case here.

Isolate Phase: Assume that again we would like to isolate the polynomial $g_\gamma(x) = x^d$. Multiplying the vector γ with the public key P yields

$$\begin{aligned}\gamma P = \Big(&\phi(a_0y_0), \phi(\beta^{s+d}a_0y_0), \phi(\beta^{2(s+d)}a_0y_0) \ldots, \phi(\beta^{(\ell-1)(s+d)}a_0y_0),\\ &\phi(a_1y_1), \phi(\beta^{s+d}a_1y_1), \phi(\beta^{2(s+d)}a_1y_1) \ldots, \phi(\beta^{(\ell-1)(s+d)}a_1y_1),\\ &\vdots \quad \vdots\\ &\phi(a_{b-1}y_{b-1}), \phi(\beta^{s+d}a_{b-1}y_{b-1}), \ldots, \phi(\beta^{(\ell-1)(s+d)}a_{b-1}y_{b-1})\Big).\end{aligned}$$

However, as β is not in the subfield we cannot continue as before. Instead $(\gamma P)^{(0)}$ and $(\gamma P)^{(1)}$ allow to recover a_0y_0 by means of $(\gamma P)^{(0)} = \phi(a_0y_0)$ and $(\gamma P)^{(1)} = \phi(\beta^{s+d}a_0y_0)$ using Equation (2), which reveals a_0y_0 as

$$a_0y_0 = \frac{(\gamma P)^{(0)}\overline{\beta^{s+d}} + (\gamma P)^{(1)}}{\beta^{s+d}+1}.$$

The same argument reveals a_jy_j using $(\gamma P)^{(j\ell)}$ and $(\gamma P)^{(j\ell+1)}$. Therefore, when looking for γ corresponding to x^d we can solve for all γ such that γP fulfils

$$(\gamma P)^{(j\ell+i)} = \phi\left(\beta^{i(s+d)}\frac{(\gamma P)^{(j\ell)}\overline{\beta^{s+d}} + (\gamma P)^{(j\ell+1)}}{\beta^{s+d}+1}\right) \tag{21}$$

for $0 \le j < b$ and $0 \le i < \ell$. We denote by Γ_d the space of all possible solutions, i.e.

$$\Gamma_d = \{\gamma \mid \gamma P \text{ fulfils Equation (21) }\}$$

Experimental Observation 8 *The dimension of Γ_d is in $\{4, 6, 8\}$.*

We next explain those dimensions.

Lemma 4. *$\{g_\gamma \mid \gamma \in \Gamma_d\}$ contains all polynomials*

$$\alpha_0x^d + \alpha_1^{d+\ell} + \alpha_2x^r + \alpha_4x^{r+\ell}$$

of degree at most $t-1$ where $r = q(d+s) - s \bmod \ell$.

Proof. For this we first claim that any polynomial fulfilling either $g(\beta x) = \beta^d g(x)$ or $\beta^s g(\beta x) = \overline{\beta^{d+s}}g(x)$ is in the set. The first condition is obvious and the second follows from the fact that in this case (using Equation (1))

$$\phi(\beta^s g(\beta x)) = \phi(\overline{\beta^{d+s}}g(x)) = \phi(\beta^{d+s}\overline{g(x)})$$

and

$$\phi(g(x)) = \phi(\overline{g(x)}).$$

If $g(x)$ is a monomial $g(x) = x^r$ we get

$$g(\beta x) = \beta^r g(x)$$

Thus, to fulfil the second equations r has to fulfil.

$$r = q(d+s) - s \bmod \ell$$

□

Clearly, the smaller the dimension of Γ_d is, the better the attack. We pick only those d such that $\dim \Gamma_d = 4$ (avoiding the exponents $d+\ell$ and $r+\ell$). The condition for this is

$$t - \ell \leq d \leq \ell \text{ and } r - \ell \leq d \leq \ell$$

and $\beta^{d+s} \notin \mathbb{F}_q$. In this case

$$\{g_\gamma \mid \gamma \in \Gamma_d\} = \{\alpha_0 x^d + \alpha_1 x^r\}$$

where $r = q(d+s) - s \bmod \ell$. For parameter set V, we ran through all possible values s and verified that in any case the number of suitable exponents d is at least 8.

Collect Phase: The collect phase, too, is different in this case. Denote by $\gamma_{(d)1}$, $\gamma_{(d)2}$ two linearly independent elements in Γ_d. Define

$$g_{\gamma_{(d)1}} = \alpha_0 x^d + \alpha_1 x^r$$

and

$$g_{\gamma_{(d)2}} = \alpha_0' x^d + \alpha_1' x^r.$$

We have

$$\begin{aligned}(\gamma_{(d)1} P)^{(i\ell)} &= \phi(a_i g(y_i)) \\ &= \phi(a_i(\alpha_0 y_i^d + \alpha_1 y_i^r)) \\ &= \phi(a_i \alpha_0 y_i^d + \overline{a_i \alpha_1 y_i^r})\end{aligned}$$

and

$$\begin{aligned}(\gamma_{(d)1} P)^{(i\ell+1)} &= \phi(a_i \beta^s g(\beta y_i)) = \phi(a_i \beta^s(\alpha_0 \beta^d y_i^d + \alpha_1 \beta^r y_i^r)) \\ &= \phi(\beta^{s+d} a_i \alpha_0 y_i^d + \overline{\beta^{s+r} a_i \alpha_1 y_i^r})) \\ &= \phi(\beta^{s+d}(a_i \alpha_0 y_i^d + \overline{a_i \alpha_1 y_i^r}))\end{aligned}$$

where we made use of the identity $\overline{\beta^{s+r}} = \beta^{s+d}$. Thus, given $(\gamma_{(d)1} P)^{(i\ell)}$ and $(\gamma_{(d)1} P)^{(i\ell+1)}$ allows us to compute

$$\eta_i = a_i \alpha_0 y_i^d + \overline{a_i \alpha_1 y_i^r}$$

and similarly

$$\eta_i' = a_i \alpha_0' y_i^d + \overline{a_i \alpha_1' y_i^r}.$$

We obtain vectors $\eta, \eta' \in \mathbb{F}_{q^2}^b$ such that

$$\begin{pmatrix} \eta \\ \eta' \end{pmatrix} = \begin{pmatrix} \alpha_0 & \overline{\alpha_1} \\ \alpha_0' & \overline{\alpha_1'} \end{pmatrix} \begin{pmatrix} a_0 y_0^d, a_1 y_1^d, \ldots, a_{b-1} y_{b-1}^d \\ \overline{a_0 y_0^r}, \overline{a_1 y_1^r}, \ldots, \overline{a_{b-1} y_{b-1}^r} \end{pmatrix}$$

Stated differently, there exist elements $\beta_0, \beta_1, \beta_2, \beta_3$ such that

$$\begin{pmatrix} \beta_0 & \beta_1 \\ \beta_2 & \beta_3 \end{pmatrix} \begin{pmatrix} \eta \\ \eta' \end{pmatrix} = \begin{pmatrix} a_0 y_0^d, a_1 y_1^d, \ldots, a_{b-1} y_{b-1}^d \\ \overline{a_0 y_0^r}, \overline{a_1 y_1^r}, \ldots, \overline{a_{b-1} y_{b-1}^r} \end{pmatrix}. \tag{22}$$

Solve Phase: We only consider the first row of Equation (22). In other words

$$\beta_0 \eta^{(i)} + \beta_1 \eta'^{(i)} = a_i y_i^d.$$

Again, we assume wlog that $a_0 = y_0 = 1$ and this allows us to represent β_1 in terms of the unknown β_0. Thus, we finally get equations

$$\beta_0 \eta^{(i)} + \left(\frac{\beta_0 \eta^{(0)} + 1}{\eta'^0} \right) \eta'^{(i)} = a_i y_i^d.$$

Using the computer algebra package MAGMA this system of equations can be solved very quickly on a standard PC. We give the running time in Table 1.