

# An Algebraic Attack Against McEliece-like Cryptosystems Based on BCH Codes

Freja Elbro and Christian Majenz  
Technical University of Denmark, Denmark;  
{freel, chmaj}@dtu.dk

**Abstract**—We present an algebraic attack on a McEliece-like scheme based on BCH codes (BCH-McEliece), where the Goppa code is replaced by a suitably permuted BCH code. Our attack continues the line of work devising attacks against McEliece-like schemes with Goppa-like codes, with the goal of getting a better understanding of why Goppa codes are so intractable. Our starting point is the work of Faugère, Perret and Portzamparc (Asiacrypt 2014). We take their algebraic model and adapt and improve their attack algorithm so that it can handle BCH-McEliece. We implement the attack and exhibit a parameter range where our attack is practical while generic attacks suggest cryptographic security.

## I. INTRODUCTION

The design of quantum-safe public key encryption schemes is a task of critical importance. This importance is highlighted by the recent NIST process to standardize quantum-resistant public-key cryptographic algorithms. One of the oldest cryptosystems not affected by known quantum computer attacks is the McEliece public key encryption scheme [1], a code-based cryptosystem based on binary Goppa codes. The main advantage of this cryptosystem is its very fast encryption/decryption, and the fact that despite decades of research, no attacks have been developed that exploit the structure of the Goppa codes. Generic decoding algorithms thus still facilitate the best-known attacks. Furthermore, despite many improvements to these generic decoding algorithms, the generic decoder from before McEliece’s time (due to Prange [2]) still has the best asymptotic runtime [3].

Although the McEliece cryptosystem has withstood cryptanalysis for almost 45 years, its security raises fundamental questions about the difficulty of recovering the secret algebraic structure of a (binary) Goppa code solely from an arbitrary basis. This is all the more intriguing as Goppa codes are linear subspaces of generalized Reed-Solomon codes for which it is famously easy to recover the algebraic structure from any basis [4]. This surprising constellation motivate the study of mathematical approaches to attacking McEliece-like schemes.

Algebraic cryptanalysis is a framework that can in principle be applied to any cryptographic scheme. It has been used successfully against several stream ciphers and cryptographic schemes based on the rank-decoding problem [5], [6]. The idea is to associate to a cryptographic primitive a set of polynomial equations such that the solutions of this system allow the recovery of secret information (like the secret key).

It was shown in [7] that it is possible to construct a polynomial system of equations for the McEliece cryptosystem

that a private key has to satisfy. This algebraic modeling works for McEliece-like cryptosystem as long as the underlying codes are alternant codes (of which Goppa codes form a subclass). In general, the resulting polynomial system has equations of high degree and/or many variables, rendering it intractable. However, it has led to attacks when alternant codes with symmetries like cyclicity or dyadicity are used [8], [9]. Here, the symmetries can be used to reduce the number of variables, making the polynomial system tractable.

More recently an attack on the cryptosystem Wild McEliece (a McEliece variant with wild Goppa codes) has been developed using the same algebraic modeling [10]. Here, the problem of high degree equations and many variables is solved in another way: exploiting multiplicities in the Goppa polynomial, more polynomial equations can be added. Clever manipulations then allow halving the number of variables of the system. For this particular system, it turns out that using only equations of low degree extends the solution set to a union of vector spaces with a nice structure. This makes it possible to find the secret key, given a basis for these vector spaces. Such a basis can be found by solving the system with low degree polynomials multiple times, each time fixing a number of variables equal to the dimension of the vector space.

The aim of this paper is to expand on the results in [10] to devise an attack on BCH-McEliece. This case is interesting for the following reasons:

- 1) The polynomial system associated to BCH codes by [7] is simpler, and hence might provide some insight, possibly also on the security of (standard) McEliece.
- 2) Narrow-sense BCH codes are a class of Goppa codes
- 3) To the best of our knowledge, there is no known better-than-generic attack against BCH-McEliece.

### A. Our contribution

We present a key recovery attack against the BCH-McEliece encryption scheme, obtained by adapting and improving the attack methodology developed in [10]. To the best of our knowledge, this is the first algebraic attack against BCH-McEliece. We provide an implementation of the attack (in Magma [11]). Experimental results obtained with this implementation show that it is practical for parts of the parameter space where generic attacks (information-set decoding) suggest cryptographic security.

On the technical side

- 1) we demonstrate that we can take the technique of [10] a step further by reducing the degrees of the polynomials in the system even further. We thus increase the dimension of the vector spaces in the solution set and thereby increase number of variables we can fix compared to [10].
- 2) we replace an exhaustive search step in the attack algorithm of [10] by an algorithm which builds the output one element at a time, improving the efficiency of the attack.

## B. Organization.

### II. NOTATION

The symbol  $\triangleq$  is used to define the left-hand side object.  $\mathbb{F}_q$  is the field with  $q$  elements where  $q$  is a power of a prime integer. When  $a \leq b$  are integers,  $\llbracket a, b \rrbracket$  is the integer interval  $[a, b] \cap \mathbb{Z}$ . When  $b < a$ ,  $\llbracket a, b \rrbracket \triangleq \emptyset$ .  $\mathbb{Z}_n \triangleq \mathbb{Z}/n\mathbb{Z}$  denotes the ring  $\{0, 1, \dots, n-1\}$  with addition and multiplication modulo  $n$ . Vectors  $\mathbf{a} = (a_1, \dots, a_n)$  will be regarded as row vectors by default. The (Hamming) weight of  $\mathbf{a} \in \mathbb{F}^n$  denoted by  $\text{wt}_H(\mathbf{a})$  is the number of non-zero coordinates. For a function  $f: \mathbb{F} \rightarrow \mathbb{F}$ , we write  $f(\mathbf{a}) \triangleq (f(a_1), \dots, f(a_n))$ . Given two fields  $\mathbb{K} \subseteq \mathbb{F}$  and  $\mathbf{A} \in \mathbb{F}^{r \times n}$ , we define

$$\ker_{\mathbb{K}} \mathbf{A} \triangleq \{\mathbf{v} \in \mathbb{K}^n \mid \mathbf{A}\mathbf{v}^T = \mathbf{0}\}.$$

$\mathbb{K}[X]$  is the ring of univariate polynomials with coefficients in the field  $\mathbb{K}$ . We also denote the ring of multivariate polynomials over  $\mathbb{K}$  with variables  $\mathbf{X} = (X_1, \dots, X_n)$  by  $\mathbb{K}[\mathbf{X}]$ . Given  $\mathcal{P} = \{P_1, \dots, P_k\} \subseteq \mathbb{K}[\mathbf{X}]$  and a field  $\mathbb{F} \supseteq \mathbb{K}$ , we define

$$\mathcal{Z}_{\mathbb{F}}(\mathcal{P}) \triangleq \{\mathbf{x} \in \mathbb{F}^n \mid \forall i \in \llbracket 1, k \rrbracket, P_i(\mathbf{x}) = 0\}.$$

Given two  $n$ -tuples  $\mathbf{x}$  and  $\mathbf{y}$  from  $\mathbb{F}^n$ , we define the Vandermonde matrix  $\mathbf{V}_t(\mathbf{x}, \mathbf{y}) \in \mathbb{F}^{t \times n}$  of order  $t \in \mathbb{N}$

$$\mathbf{V}_t(\mathbf{x}, \mathbf{y}) \triangleq \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ y_1 x_1 & y_2 x_2 & \cdots & y_n x_n \\ \vdots & \vdots & \ddots & \vdots \\ y_1 x_1^{t-1} & y_2 x_2^{t-1} & \cdots & y_n x_n^{t-1} \end{bmatrix}.$$

### III. ALGEBRAIC CODING THEORY

This section is devoted to recalling classical notions on algebraic coding theory. We refer to [12] for a detailed treatment.

A linear code  $\mathcal{C}$  of length  $n$  and dimension  $k$  over a field  $\mathbb{F}$  is a  $k$ -dimensional subspace of  $\mathbb{F}^n$  where  $k < n$  are natural numbers. Any matrix such that its rows form a basis of  $\mathcal{C}$  is called a generator matrix. A generator matrix  $\mathbf{G}$  is said to be on standard form if  $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$  with  $\mathbf{A} \in \mathbb{F}^{k \times n-k}$ . The dual of  $\mathcal{C} \subseteq \mathbb{F}^n$  is the linear space

$$\mathcal{C}^\perp \triangleq \left\{ \mathbf{z} \in \mathbb{F}^n \mid \forall \mathbf{u} \in \mathcal{C}, \sum_i u_i v_i = 0 \right\}.$$

We always have  $\dim \mathcal{C}^\perp = n - \dim \mathcal{C}$ , and a generator matrix of  $\mathcal{C}^\perp$  is called a parity check matrix of  $\mathcal{C}$ . Moreover, if  $[\mathbf{I}_k \mid \mathbf{A}]$  is a generator matrix for  $\mathcal{C}$  then  $[\mathbf{A}^T \mid \mathbf{I}_{n-k}]$

is a parity-check matrix for  $\mathcal{C}$ . An algorithm  $\Phi$  is  $t$ -correcting if  $\Phi(\mathbf{u} + \mathbf{e}) = \mathbf{u}$  for every  $(\mathbf{u}, \mathbf{e}) \in \mathcal{C} \times \mathbb{F}^n$  when  $\text{wt}_H(\mathbf{e}) \leq t$ .

We now recall definitions and key properties of the linear codes, which are relevant for this article.

**Definition 1** (Generalized Reed-Solomon code). *Let  $k$  and  $n$  be integers such that  $1 \leq k < n \leq q$  where  $q$  is a power of a prime number. The generalized Reed-Solomon code  $\text{GRS}_k(\mathbf{x}, \mathbf{y})$  of dimension  $k$ , where  $\mathbf{x}$  is an  $n$ -tuple of distinct elements of  $\mathbb{F}_q$  and  $\mathbf{y}$  is an  $n$ -tuple of nonzero elements in  $\mathbb{F}_q$ , is defined as the code with generator matrix  $\mathbf{V}_k(\mathbf{x}, \mathbf{y})$ .*

One of the key properties of GRS-codes is that they can be efficiently decoded.

**Proposition 2.** *For every generalized Reed-Solomon code  $\text{GRS}_k(\mathbf{x}, \mathbf{y})$  there exists a polynomial-time algorithm that corrects all errors of weight at most  $(n-k)/2$  whenever  $\mathbf{x}$  and  $\mathbf{y}$  are known. We denote this decoder by  $\text{decGRS}_{k,\mathbf{x},\mathbf{y}}$*

Another key property is that the dual of a GRS-code is again a GRS-code.

**Proposition 3.**  $\text{GRS}_k(\mathbf{x}, \mathbf{y})^\perp = \text{GRS}_{n-k}(\mathbf{x}, \mathbf{z})$  where for every  $i \in \llbracket 1, n \rrbracket$ ,

$$z_i^{-1} \triangleq y_i \prod_{j=1, j \neq i}^n (x_j - x_i)$$

We turn now to subfield subcodes of GRS-codes.

**Definition 4** (Alternant code). *Given a natural number  $m > 1$  the  $q$ -ary alternant code  $\mathcal{A}_r(\mathbf{x}, \mathbf{y})$ , where  $\mathbf{x}$  is an  $n$ -tuple of distinct elements of  $\mathbb{F}_{q^m}$  and  $\mathbf{y}$  is an  $n$ -tuple of nonzero elements in  $\mathbb{F}_{q^m}$ , is the following (subfield) subcode*

$$\mathcal{A}_r(\mathbf{x}, \mathbf{y}) = \text{GRS}_r(\mathbf{x}, \mathbf{y})^\perp \cap \mathbb{F}_q^n$$

*Remark 5.* We note that  $\mathcal{A}_r(\mathbf{x}, \mathbf{y}) \subset \text{GRS}_{n-r}(\mathbf{x}, \mathbf{z})$  with  $\mathbf{z}$  defined in proposition 3, so  $\text{decGRS}_{n-r,\mathbf{x},\mathbf{z}}$  can be used as a decoding algorithm for  $\mathcal{A}_r(\mathbf{x}, \mathbf{y})$ . It can correct up to  $r/2$  errors whenever  $\mathbf{x}$  and  $\mathbf{z}$  are known – or equivalently whenever  $\mathbf{x}$  and  $\mathbf{y}$  are known.

Another key property of alternant codes is the following:

**Corollary 6.** *Given  $\mathbf{x}$  that is an  $n$ -tuple of distinct elements of  $\mathbb{F}_{q^m}$  and  $\mathbf{y}$  which is an  $n$ -tuple of nonzero elements in  $\mathbb{F}_{q^m}$ , it holds that*

$$\mathcal{A}_r(\mathbf{x}, \mathbf{y}) = \ker_{\mathbb{F}_q} \mathbf{V}_r(\mathbf{x}, \mathbf{y}).$$

We turn now to specific alternant codes.

**Definition 7** (Goppa code). *A  $q$ -ary Goppa code  $\mathcal{G}(\mathbf{x}, \gamma)$  of length  $n$ , where  $\mathbf{x}$  is an  $n$ -tuple of distinct elements from  $\mathbb{F}_{q^m}$  with  $m > 1$  and  $\gamma$  is a polynomial in  $\mathbb{F}_{q^m}[X]$  of degree  $t$  such that  $\mathbf{x}$  does not contain any root of  $\gamma$ , is the alternant code  $\mathcal{A}_t(\mathbf{x}, \mathbf{y})$  where  $y_j \triangleq \gamma(x_j)^{-1}$  for every  $j \in \llbracket 1, n \rrbracket$ .*

A key property of Goppa-codes is the following:

**Proposition 8** ([13]). *Let  $\gamma \in \mathbb{F}_{q^m}[X]$  be a monic and square free polynomial of degree  $t$ . Let  $\mathbf{x}$  be an  $n$ -tuple of distinct*

elements of  $\mathbb{F}_{q^m}$  satisfying  $\gamma(x_i) \neq 0$  for every  $i \in \llbracket 1, n \rrbracket$ . Then

$$\mathcal{G}(\mathbf{x}, \gamma^{q-1}) = \mathcal{G}(\mathbf{x}, \gamma^q).$$

The last code-class we introduce, is the BCH-codes, which are also a subclass of alternant codes.

**Definition 9** (BCH code). *Let us assume that  $q = p^s$  where  $p$  is a prime number and  $s \geq 1$ . Let  $b, m$  and  $n$  be a natural numbers such that  $n$  divides  $q^m - 1$ . Let  $\alpha$  be an arbitrary primitive  $n$ -th root of unity in  $\mathbb{F}_{q^m}$ .*

*A  $q$ -ary BCH code, of length  $n$  denoted by  $\text{BCH}_{b,r}(\alpha)$  is the  $q$ -ary alternant code  $\mathcal{A}_r(\alpha, \alpha^b)$ , where  $\alpha \triangleq (1, \alpha, \alpha^2, \dots, \alpha^{n-1})$ .*

*When  $b = 1$  the BCH-code is called narrow-sense and when  $n = q^m - 1$  it is called primitive.*

It is more standard to define a BCH-code in terms of its designed distance  $\delta$ . It is then defined as the cyclic code with zeros  $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$ . If we set  $\delta = r + 1$ , then the two definitions are equivalent. We chose definition 9, as all the properties of BCH-codes that will be relevant to us follow directly from it and we want to emphasize the connection to Goppa codes.

**Proposition 10.** *A narrow-sense  $q$ -ary BCH code  $\text{BCH}_{1,r}(\alpha)$  is the  $q$ -ary Goppa code  $\mathcal{G}(\alpha^{-1}, X^r)$  where  $\alpha^{-1} = (1, \alpha^{-1}, \dots, \alpha^{-(n-1)})$ .*

*Proof.* By definition of a narrow-sense BCH code, any codeword is in the right kernel of the following matrix

$$\begin{bmatrix} 1 & \alpha^1 & \alpha^2 & \dots & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{(n-1)2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^r & \alpha^{2r} & \dots & \alpha^{(n-1)r} \end{bmatrix}. \quad (1)$$

For every  $i \in \llbracket 1, r \rrbracket$  and  $j \in \llbracket 1, n \rrbracket$  the entry located at position  $(i, j)$  in the matrix (1) is equal to  $\alpha^{i(j-1)}$ . But by letting  $\ell \triangleq r - i$  and setting  $a_j \triangleq \alpha^{-(j-1)}$  we then have

$$\begin{aligned} \alpha^{i(j-1)} &= \alpha^{(r-\ell)(j-1)} = \alpha^{(r)(j-1)} \alpha^{-\ell(j-1)} \\ &= a_j^{-(r)} a_j^\ell = \frac{a_j^\ell}{a_j^r} \end{aligned}$$

As  $\ell \in \llbracket 0, r-1 \rrbracket$  this proves that a narrow-sense BCH code is a Goppa code defined with the polynomial  $X^r$ .  $\square$

**Theorem 11** ([14]). *The dimension of a narrow-sense BCH code  $\text{BCH}_{1,r}(\alpha)$  is at least  $n - m \lceil r - r/q \rceil$ .*

#### IV. THE McELIECE CRYPTOSYSTEM

The McEliece cryptosystem [1] is a public key encryption scheme. We recall its key generation, encryption and decryption functions in algorithm 1, 2 and 3 respectively. We give the description for a generalization of the McEliece scheme based on alternant codes. To get the original McEliece cryptosystem, you just have to specialize to  $q = 2$  and  $\mathbf{y} = \gamma(\mathbf{x})^{-1}$ . Then apply proposition 8 to double the number of errors that can be corrected.

In the algorithms we will write  $x \stackrel{\$}{\leftarrow} \mathcal{S}$  to express that  $x$  is sampled according to the uniform distribution over a set  $\mathcal{S}$ .

---

**Algorithm 1**  $(\text{sk}, \text{pk}) \leftarrow \text{Gen}(\mathbb{1}^\lambda)$

---

- 1:  $(q, m, n, r) \leftarrow \text{param}(\mathbb{1}^\lambda)$
  - 2:  $\mathbf{x} \stackrel{\$}{\leftarrow} \mathbb{F}_{q^m}^n$  composed of pairwise distinct entries
  - 3:  $\mathbf{y} \stackrel{\$}{\leftarrow} (\mathbb{F}_{q^m}^*)^n$
  - 4: Pick at random a generator matrix  $\mathbf{G}$  of  $\mathcal{A}_r(\mathbf{x}, \mathbf{y})$
  - 5: **return** Return  $(\text{sk}, \text{pk})$  with  $\text{sk} \triangleq (\mathbf{x}, \mathbf{y})$  and  $\text{pk} \triangleq (\mathbf{G}, \frac{r}{2})$
- 

---

**Algorithm 2**  $\mathbf{c} \leftarrow \text{Enc}(\text{pk}, \mathbf{m})$

---

- 1: Parse  $\text{pk}$  to obtain  $\mathbf{G}, r$
  - 2:  $\mathbf{e} \stackrel{\$}{\leftarrow} \mathbb{F}_q^n$  of weight  $\text{wt}_H(\mathbf{e}) = \lfloor r/2 \rfloor$
  - 3: **return** Return  $\mathbf{c} \triangleq \mathbf{m}\mathbf{G} + \mathbf{e}$
- 

---

**Algorithm 3**  $\mathbf{m} \leftarrow \text{Dec}(\text{sk}, \mathbf{c})$

---

- 1: Parse  $\text{sk}$  to get  $\mathbf{x}$  and  $\mathbf{y}$
  - 2: Define  $\mathbf{z}$  as in prop. 3 from  $\mathbf{x}$  and  $\mathbf{y}$
  - 3: **return** Return  $\mathbf{m} \triangleq \text{decGRS}_{n-r, \mathbf{x}, \mathbf{z}}(\mathbf{c})$
- 

#### A. Algebraic Modeling of the Key Recovery Problem

We here explain how recovering the secret key of any McEliece-like cryptosystem based on alternant codes is connected to solving a polynomial system. The adversary has access to the (public) generator matrix  $\mathbf{G} = (g_{i,j})$  belonging to  $\mathbb{F}_q^{k \times n}$ . This matrix contains a basis for an alternant code  $\mathcal{A}_r(\mathbf{x}, \mathbf{y})$  of length  $n \leq q^m$  and dimension  $k \geq n - rm$ .

The algebraic modeling given in [7] introduces  $2n$  variables  $\mathbf{X} \triangleq (X_1, \dots, X_n)$  and  $\mathbf{Y} \triangleq (Y_1, \dots, Y_n)$  satisfying the following,

$$\sum_{j=1}^n g_{i,j} Y_j X_j^u = 0, \quad 1 \leq i \leq k, \quad 0 \leq u \leq r-1, \quad (2)$$

Substituting  $\mathbf{X} \leftarrow \mathbf{x}$  and  $\mathbf{Y} \leftarrow \mathbf{y}$  into the system yields a valid solution, because

$$\mathcal{A}_r(\mathbf{x}, \mathbf{y}) = \ker_{\mathbb{F}_q} \mathbf{V}_r(\mathbf{x}, \mathbf{y}),$$

see Cor. 6.

As Goppa codes are alternant codes, we can use the same modeling for them. We do have the additional information that  $\mathbf{y} = \gamma(\mathbf{x})^{-1}$ , but we do not know of a way to leverage this information without increasing massively the degree of the equations. So, like [7], we do not give a separate modeling for Goppa codes.

A classical approach for solving such a system is to use algorithms relying on the computation of Gröbner bases. But in a cryptographic setting, the number of unknowns can be really high, as  $n$  is larger than 1,000. Unless the number of unknowns is very low (less than 100 as in [7]), no existing algorithms can deal with the current cryptographic parameters.

These practical obstacles do not change the fact that key recovery problem of the McEliece public-key cryptosystem has a strong connection to the following problem

**Problem 1.** *Considering the polynomial ring  $\mathbb{F}_q[\mathbf{X}, \mathbf{Y}]$  with  $\mathbf{X} \triangleq (X_1, \dots, X_n)$  and  $\mathbf{Y} \triangleq (Y_1, \dots, Y_n)$ . Compute  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P})$  where*

$$\mathcal{P} \triangleq \left\{ \sum_{j=1}^n g_{i,j} Y_j X_j^\ell \mid i \in \llbracket 1, k \rrbracket, \ell \in \llbracket 0, r-1 \rrbracket \right\} \quad (3)$$

and  $\mathbf{G} = (g_{i,j}) \in \mathbb{F}_q^{k \times n}$  is a generator matrix of a  $q$ -ary alternant code  $\mathcal{A}_r(\mathbf{x}, \mathbf{y}) \subset \mathbb{F}_q^n$  of dimension  $k$ .

However, the two are not equivalent, as  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P})$  might contain more than just the secret key  $(\mathbf{x}, \mathbf{y})$ . In fact, we know that the solution set contains at least  $m$  elements as the zeros are sought in  $\mathbb{F}_{q^m}$ , but each polynomial in  $\mathcal{P}$  has its coefficients in  $\mathbb{F}_q$ . Hence we can apply the Frobenius mapping defined for  $z \in \mathbb{F}_{q^m}$  by  $z \mapsto z^{q^e}$  with  $e \in \llbracket 0, m-1 \rrbracket$  to get other solutions in  $\mathbb{F}_{q^m}$ .

**Lemma 12.** *If  $(\mathbf{x}, \mathbf{y})$  is in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P})$ , then so is  $(\mathbf{x}^{q^e}, \mathbf{y}^{q^e})$  for every  $e \in \llbracket 1, m-1 \rrbracket$ .*

Furthermore, Theorem 4 in [15] shows that even though  $(\mathbf{x}', \mathbf{y}') \neq (\mathbf{x}, \mathbf{y})$ , it might still be the case that  $\text{GRS}_k(\mathbf{x}', \mathbf{y}') = \text{GRS}_k(\mathbf{x}, \mathbf{y})$ . If this is the case, then  $(\mathbf{x}', \mathbf{y}')$  is also in the zero set of  $\mathcal{P}$ .

These two groups of extra solutions are not a problem though, as they all generate the same alternant code, so regardless of which one you choose, you can get a decoding algorithm for the code generated by  $G$ .

However,  $\mathcal{P}$  also has zeros which does not generate the same alternant code. In fact, these extra solutions often do not generate an alternant code at all, as they do not consist of distinct/non-zero elements. These solutions are not well characterized, but it has been found experimentally that if we include extra polynomials which encode the information that  $\mathbf{x}$  must consist of  $n$  distinct elements and all entries of  $\mathbf{y}$  must be non-zero, then the solution set becomes manageable, so finding the secret key becomes equivalent to solving problem 1 with the inclusion of those extra polynomials [7]. The drawback of this approach is of course the extra variables added to an already intractable system.

## V. CRYPTANALYSIS OF THE BCH-MCELIECE CRYPTOSYSTEM

### A. Preliminaries

In this section we describe a key-recovery attack on the BCH-McEliece cryptosystem, which uses permuted narrow-sense primitive BCH codes. We will introduce additional notation throughout this section, and therefore provide an overview of notation in appendix A. We hope that this can serve as an index in case our readers want to go back and look up a definition.

Algorithm 8 describes the key generation process of the BCH-McEliece cryptosystem. The encryption and decryption

functions do not change from the previous section, so they can be found in Algorithm 2 and Algorithm 3 respectively.

We recall from Prop. 10 that a narrow-sense BCH code is the Goppa code  $\mathcal{G}(\alpha, X^r)$  where  $\alpha \triangleq (1, \alpha^{-1}, \dots, \alpha^{-(n-1)})$ .

---

**Algorithm 4**  $(\text{sk}, \text{pk}) \leftarrow \text{Gen}(\mathbb{1}^\lambda)$

---

- 1:  $(p, s, m, r) \leftarrow \text{param}(\mathbb{1}^\lambda)$
  - 2:  $q \leftarrow p^s$
  - 3:  $n \leftarrow q^m - 1$
  - 4: Pick at random in  $\mathbb{F}_{q^m}$  a primitive  $n$ -th root of unity  $\alpha$
  - 5: Pick at random a permutation  $\pi$  on  $\llbracket 0, n-1 \rrbracket$
  - 6:  $\mathbf{x} \leftarrow (\alpha^{\pi(0)}, \alpha^{\pi(-1)}, \dots, \alpha^{\pi(-(n-1))})$
  - 7: Pick at random a generator matrix  $\mathbf{G}$  of  $\mathcal{G}(\mathbf{x}, X^r)$
  - 8: **return**  $(\text{sk}, \text{pk})$  with  $\text{sk} \triangleq \mathbf{x}$  and  $\text{pk} \triangleq (\mathbf{G}, r/2)$
- 

The key recovery problem we want to solve is

**Problem 2.** *Given a public key  $\mathbf{G}$  that is obtained from the BCH-McEliece cryptosystem instantiated with a  $k$ -dimensional  $q$ -ary permuted narrow-sense primitive BCH code  $\text{BCH}_r(\mathbf{x})$ . Assuming that  $q = p^s$  where  $p$  is a prime number and  $s \geq 1$ , find  $\mathbf{x}$ .*

Following [10], we will only use equations of low degree in the algebraic modeling of [7]. This makes the solution set of the system larger, naively increasing the difficulty of the remaining search problem. However, the extra solutions are helpful as they form a vector space with a very particular structure. We can therefore 1) repeatedly fix variables when solving the system, which means that it is easier to solve, and 2) “detangle” the vector spaces to obtain the correct solution.

### B. Algebraic modeling

We specialize the algebraic modeling of [7] to the BCH-case and thus see that  $\mathbf{x}$  is a zero of

$$\mathcal{P} \triangleq \left\{ \sum_{j=1}^n g_{i,j} X_j^\ell \mid i \in \llbracket 1, k \rrbracket, \ell \in \llbracket 1, r \rrbracket \right\}. \quad (4)$$

We can thus solve Problem 2 via finding  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P})$ .

### C. Focusing on a low degree polynomial system.

To obtain a suitable restriction of  $\mathcal{P}$  to low-degree polynomials, we first observe that  $\mathcal{P}$  contains  $k$  linear polynomials. Following [10], we further observe that for every  $a \in \llbracket 0, sm \rrbracket$ , the mapping from  $\mathbb{F}_{q^m}$  to  $\mathbb{F}_{q^m}$  given by  $z \mapsto z^{p^a}$  is  $\mathbb{F}_p$ -linear and invertible. This enables us to find more linear equations that  $\mathbf{x}$  satisfies. If we define  $\sqrt[p^a]{\cdot} : \mathbb{F}_{q^m} \rightarrow \mathbb{F}_{q^m}$  to be the inverse of  $z \mapsto z^{p^a}$ , then the following is straight forward to prove

**Lemma 13.** *For any  $a \in \llbracket 0, sm \rrbracket$  such that  $p^a \in \llbracket 1, r \rrbracket$ , and for every  $i \in \llbracket 1, k \rrbracket$  it holds that*

$$\sum_{j=1}^n \sqrt[p^a]{g_{i,j}} x_j = 0.$$



The secret key  $\mathbf{x}$  is therefore additionally a zero of

$$\mathcal{L} \triangleq \left\{ \sum_{j=1}^n \sqrt[p^a]{g_{i,j}} X_j \mid i \in \llbracket 1, k \rrbracket, p^a \in \llbracket 1, r \rrbracket \right\}. \quad (5)$$

Solving linear equations is fast, but unfortunately we were not able to detangle  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{L})$  to obtain the secret key. However, we found that adding the following equations

$$\mathcal{P}_u \triangleq \left\{ \sum_{j=1}^n g_{i,j} X_j^u \mid i \in \llbracket 1, k \rrbracket \right\}. \quad (6)$$

where

$$u \triangleq \begin{cases} 3 & \text{for } p = 2 \\ 2 & \text{else} \end{cases}$$

gave a zero-set that we could detangle.<sup>1</sup>

An outline of our attack is as follows: We solve Problem 2 by identifying a particular vector space  $V \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  such that  $\mathbf{x} \in V$  and such that it is possible to easily recover  $\mathbf{x}$  from the knowledge of  $V$ .

Finding a basis for  $V$  is much easier than finding zeros of  $\mathcal{P}$  directly. In part because we can fix  $\dim(V)$  many variables when we solve the system, and in part because we only have polynomials of low degree in our system. This means that we can break the BCH-McEliece cryptosystem for many parameters. See Section VII

#### D. Identifying a vector space in $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ .

An important feature of the attack given in [10] is to identify a relatively large vector space that is included in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ . The idea is to find sets  $L \subseteq \mathbb{Z}_{q^m-1}$  which guarantees that

$$\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \} \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L}).$$

We now introduce a conditions that will help us to find such sets  $L$ . In the following,  $\mathcal{C}_q(c)$  are the cyclotomic cosets,

$$\mathcal{C}_q(c) \triangleq \{ cq^e \pmod{q^m-1} \mid e \in \llbracket 0, m-1 \rrbracket \},$$

and  $\mathcal{C}_q \triangleq \bigcup_{c \in \llbracket 1, r \rrbracket} \mathcal{C}_q(c)$  is their union. Note that the same reasoning as in lemma 12,  $\bigcup_{c \in \llbracket 1, r \rrbracket} \mathcal{C}_q(c)$  contains powers  $c$  for which we know that

$$\sum_{j=1}^n g_{i,j} x_j^c = 0 \text{ for all } i \in \llbracket 1, r \rrbracket. \quad (7)$$

**Definition 14.** For  $u \in \llbracket 1, r \rrbracket$ ,  $L \subseteq \mathbb{Z}_{q^m-1}$  is said to satisfy the  $u$ -sum condition, if

- U1: for every  $\ell \in L$  and any  $p^a \in \llbracket 1, r \rrbracket$ ,  $p^a \cdot \ell \in \mathcal{C}_q$ , and  
 U2: for any choice of  $\ell_1, \dots, \ell_u \in L$ ,  $\ell_1 + \dots + \ell_u \in \mathcal{C}_q$ .

<sup>1</sup>The systems of [10] includes more non-linear equations than ours. There is a trade-off between the benefit of more equations and the dimension of the vector spaces in the solution set. Both our experiments and those of [10] indicate that it is more important to raise the dimension of the solution set than it is to keep more equations. We take that idea to its logical conclusion, and include as few non-linear equations as possible.

**Proposition 15.** If  $L \subseteq \mathbb{Z}_{q^m-1}$  satisfies the  $u$ -sum condition then

$$\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \} \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L}).$$

*Proof.* We will prove the more precise statements:

- 1) U1  $\Rightarrow \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \} \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{L})$ .
- 2) U2  $\Rightarrow \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \} \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u)$ .

For the first of these: Let  $\mathbf{z} \in \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \}$  be given for some  $L$ , which satisfies the U1 condition. Then

$$\mathbf{z} = \sum_{\ell \in L} c_\ell \mathbf{x}^\ell \quad (8)$$

for some  $c_\ell$ 's in  $\mathbb{F}_{q^m}$ . We now have to prove that

$$\sum_{j=1}^n \sqrt[p^a]{g_{ij}} z_j = 0 \quad (9)$$

for all  $p^a \in \llbracket 1, r \rrbracket$ . But since raising to the power  $p^a$  is  $\mathbb{F}_p$ -linear for all elements in  $\mathbb{F}_{q^m}$ , this is equivalent to

$$\sum_{j=1}^n g_{ij} z_j^{p^a} = 0 \quad (10)$$

If we now plug in  $z_j$  from equation (8), and again use that raising to the power  $p^a$  is  $\mathbb{F}_p$ -linear, we see that

$$\sum_{j=1}^n g_{ij} z_j^{p^a} = \sum_{j=1}^n g_{ij} \sum_{\ell \in L} c_\ell^{p^a} x_j^{\ell \cdot p^a} = \sum_{\ell \in L} c_\ell^{p^a} \sum_{j=1}^n g_{ij} x_j^{\ell \cdot p^a}$$

Which is equal to zero by (7) as  $\ell \cdot p^a \in \bigcup_{c \in \llbracket 1, r \rrbracket} \mathcal{C}_q(c)$  for all  $l$  and  $p^a$ . This finishes the proof.

The proof of the second statement is similar. Except here, raising to the power  $u$  is not linear in general, so the last step becomes

$$\begin{aligned} \sum_{j=0}^n g_{ij} z_j^u &= \sum_{j=0}^n g_{ij} \sum_{\ell_1, \dots, \ell_u \in L} b_{\ell_1, \dots, \ell_u} x_j^{\ell_1 + \dots + \ell_u} \\ &= \sum_{\ell_1, \dots, \ell_u \in L} b_{\ell_1, \dots, \ell_u} \sum_{j=0}^n g_{ij} x_j^{\ell_1 + \dots + \ell_u} \\ &= 0 \end{aligned}$$

□

**Remark 16.** Note that  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  might contain more than one vector space. In fact, for most parameters, it contains many vector spaces, which may or may not have overlapping subspaces. For example for  $q = 8$ ,  $m = 2$  and  $r = 20$ , all of the following sets satisfy the 3-sum condition:

- $\{1, 2, 3, 4, 6\}$  and  $\{8, 16, 24, 32, 48\}$
- $\{1, 2, 3, 4, 8\}$  and  $\{8, 16, 24, 32, 1\}$
- $\{1, 2, 8, 16\}$

The following lemma shows that some of the sets we found in remark 16 were no coincidence

**Lemma 17.** If  $L \subseteq \mathbb{Z}_{q^m-1}$  satisfies the  $u$ -sum criteria, then  $qL$  does as well.

*Proof.* Relies on the simple observation that  $a \in \mathcal{C}_q \Rightarrow qa \in \mathcal{C}_q$ .  $\square$

For the attack, we need the size of the largest  $L$  as this tells us the number of variables we can fix. Once we fix this number of variables, we are unlikely to find solutions in the smaller vector spaces. Hence in the example of remark 16, if we fix 5 variables, then we are most likely to only find  $2m$  solutions.

For the remainder of this article, we will denote by  $L$  a subset of  $\mathbb{Z}_{q^{m-1}}$  of maximal size which fulfills the  $u$ -sum condition.

We were not able to find a closed formula for the size of  $L$ , but we developed two algorithms to determine  $L$  (next section). Furthermore, we found the following upper and lower limits:

**Lemma 18.** *Any  $L$  that fulfills the  $u$ -sum criteria has size at most  $rm$ . Conversely, the largest  $L$ 's that fulfill the  $u$ -sum criteria, have size at least  $\lfloor \frac{r}{\max U} \rfloor$ , where*

$$U \triangleq \{u\} \cup \{p^a \mid a \in \llbracket 0, s-1 \rrbracket, p^a \in \llbracket 1, r \rrbracket\}$$

*Proof.* For the first statement, note that  $L \subseteq \mathcal{C}_q$ , and  $\mathcal{C}_q$  has size at most  $rm$ . For the second, we claim that  $\{1, \dots, \lfloor \frac{r}{\max U} \rfloor\}$  fulfills the  $u$ -sum criteria. The central step of proving the claim is to realize that

$$p^a \cdot l \in \mathcal{C}_q \Rightarrow q \cdot p^a \cdot l \in \mathcal{C}_q.$$

Hence, to check U1 of definition 14 it is sufficient to check  $p^a$  where  $a \in \llbracket 0, s-1 \rrbracket$ .  $\square$

#### *E. Algorithms to find $L$ 's which fulfill the $u$ -sum criteria*

Here we present two algorithms to find  $L$ 's of maximal size that fulfill the  $u$ -sum criteria. The first algorithm is fast but may not always find an  $L$  of maximal size. The second is guaranteed to find all  $L$ 's of maximal size, but may have runtime exponential in  $r$ .

The first step for both of the algorithms is to create the set

$$S_U \triangleq \{l \in \mathbb{Z}_{q^{m-1}} \mid l \cdot u \in \mathcal{C}_q \text{ for all } u \in U\}$$

where  $U$  is defined in lemma 18. We create this set to limit the search space for  $L$ , as it is clear that  $L \subseteq S_U$ . Furthermore,  $S_U$  can be created efficiently by running through all elements  $l$  of  $\mathbb{Z}_{q^{m-1}}$  and for each test if  $l \cdot u \in \mathcal{C}_q$  for all  $u \in U$ .

In the fast algorithm (Algorithm 5), the idea is simply create  $L$  element by element. To initialize, we set  $L = \{\}$ , and then for each element  $l$  of  $S_U$ , we check if  $L \cup \{l\}$  fulfills the  $u$ -sum criteria. If it does, then we grow  $L$  and if not, then we leave out that particular element, and move onto the next. Based on our experience with how the sets look, we decided to sort the elements of  $S_U$  and start by adding small elements. We can check if a set  $L$  fulfills the  $u$ -sum criteria in time  $O(|L|^3)$  (we need to calculate a sum for each choice of two or three elements from  $L$ ). As we need to check a  $u$ -sum criteria for each  $l \in S_U$ , the total runtime of the algorithm is  $O(|S_U|^4)$ , which is  $O(m^4 r^4)$  by the same argument as in lemma 18.

---

#### **Algorithm 5** Find $L$ Fast

---

**Input:**  $S_U$  and  $U$

**Output:** A set  $L \subseteq \mathbb{Z}_{q^{m-1}}$ , which fulfills the  $u$ -sum criteria

**Algorithm:**

```

 $L := \{\}$ 
for  $l \in S_U$  do
  if  $L \cup \{l\}$  fulfills the  $u$ -sum criteria then
     $L := L \cup \{l\}$ 
  end if
end for
return  $L$ 

```

---

The second algorithm (Algorithm 6), which is guaranteed to find all  $L$ 's of maximal size, leverages the maximal clique finding algorithms from graph theory. The idea is to construct a hypergraph where each vertex is an element of  $S_U$ , and a set of vertices is connected by an edge, if the set fulfills the  $u$ -sum criteria. Once this graph is constructed, finding  $L$  boils down to finding the maximal cliques of the graph. When  $p \neq 2$ , it is sufficient to consider edges of size two (the hypergraph is just a normal graph). This is because a set  $L$  fulfills the 2-sum criteria if and only if all subsets of  $L$  of size two fulfill the criteria. For  $p = 2$ , we need to consider hypergraphs, where all edges connect three vertices.

---

#### **Algorithm 6** Find $L$ Accurately for $u = 3$

---

**Input:**  $S_U$  and  $U$

**Output:** All sets  $L \subseteq \mathbb{Z}_{q^{m-1}}$  of maximal size, which fulfill the  $u$ -sum criteria

**Algorithm:**

```

Initialize hypergraph  $\mathcal{G}$  with nodeset  $S_U$ .
for  $\{a, b, c\} \subseteq S_U$  do
  if  $\{a, b, c\}$  fulfills the  $u$ -sum criteria then
    add edge  $\{a, b, c\}$  to  $\mathcal{G}$ 
  end if
end for
Run algorithm for finding maximal cliques in  $\mathcal{G}$ 
return Maximal cliques

```

---

In our code, we implemented the fast but inaccurate algorithm for  $p = 2$ , as magma does not support hypergraphs, and we did not run into any cases where the algorithm failed. For  $p \neq 2$ , we implemented the slow but accurate algorithm, and we used the clique finding algorithm of Magma, which is based on [16]. The runtime of this algorithm is not well understood in general, but we note that it is at most exponential in the number of vertices  $|S_U|$ , i.e. in  $r$  and  $m$ .

#### *F. Repeatedly solving restrictions of the polynomial system*

We use Magma's [11] Gröbner basis algorithm to find vector spaces of maximal dimension in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  as follows. The set  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  contains at least one vector space of dimension  $|L|$ . We can thus fix  $|L|$  variables, and still find a solution in each vector space of maximal dimension. We run the Gröbner basis algorithm  $|L|$  times, each time fixing  $|L|$  variables differently, to find  $|L|$  sets of solutions.

In the first run, we fix the first  $|L|$  variables to

$$(1, \omega, \omega^2, \dots, \omega^{|L|-1})$$

where  $\omega$  is a primitive  $n$ 'th root of unity, that we choose. In the second run, we fix the variables to

$$(1, \omega^2, \omega^4, \dots, \omega^{2(|L|-1)})$$

etc. This choice of fixed variables is different from the choice of [10]. Our choice forces the solutions to have many different entries, which we found to be helpful when trying to avoid additional solutions outside of the vector spaces that we understand.

Let us denote the solution sets outputted by the Gröbner basis algorithm by  $V_1, \dots, V_{|L|}$ . If we denote by  $\nu$  the number of vector spaces of maximal dimension in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ , then with high probability,  $V_i$  has exactly  $\nu$  elements – one from each of these vector spaces. To obtain a basis for each of the maximal vector spaces, what is left to do is to “match up” the elements of  $V_i$ , i.e., to find sets  $W_j$ ,  $j = 1, \dots, \nu$  such that  $W_j \cap W_{j'} = \emptyset$  for  $j \neq j'$ ,  $|V_i \cap W_j| = 1$ , and  $\text{span}_{\mathbb{F}_{q^m}} W_j \subseteq \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ . This is achieved via Fröbenius Alignment.

#### G. Step 4: Fröbenius Alignment

In [10], the above task is solved using a brute-force search, a routine with runtime exponential in  $|L|$ . This is possible since their  $L$  is not too big. However, as  $r$  grows,  $|L|$  grows too rendering this step infeasible.

We are able to construct a faster solution by leveraging the following insight: if two vectors,  $\mathbf{v}, \mathbf{w} \in \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  belong to the same vector space of zeros, then  $\mathbf{v} + \mathbf{w}$  is also in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ , and if they don't belong to the same vector space, then most likely,  $\mathbf{v} + \mathbf{w}$  is not in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ . Hence, we can build a basis  $W$  iteratively. First we choose  $\mathbf{w}_1 \in V_1$ . Then, for each  $i$ , we check whether  $\mathbf{w}_1 + \mathbf{w}_i$  is a zero of  $\mathcal{P}_u \cup \mathcal{L}$  for all  $\mathbf{w}_i \in V_i$  and add the successful vector to  $W$ .

For each choice of  $\mathbf{w}_1$  this gives a basis for a vector space in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ . If we loop through all the different choices of  $\mathbf{w}_1$ , we find bases for all the vector spaces in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  of maximal dimension.

Note that this algorithm is not fail safe: it is possible that  $\mathbf{v} + \mathbf{w} \in \mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$ , but  $\mathbf{v}$  and  $\mathbf{w}$  are not in the same vector space of maximal dimension. There are ways to make our Fröbenius Alignment algorithm more dependable, but for all the parameters we tried, the simple version of the algorithm sufficed.

The algorithm is summarized in Algorithm 7. The runtime is  $O(|L| \cdot \nu)$ , where  $\nu$  is the number of results given by the Gröbner basis calculation.

#### H. Step 5: Detanglement

Here we assume that we have a basis for all vector spaces of maximal dimension in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  and that we know all the corresponding  $L$ 's. We further assume that we know which  $L$  and  $W$  belong together (we can achieve this by trying all

---

#### Algorithm 7 Fast Fröbenius Alignment

---

##### Inputs:

- $|L|$  lists of solutions to  $\mathcal{W}_U$ :  $V_1, \dots, V_{|L|}$
- Index  $j \in \{1, \dots, |V_1|\}$  (deciding which solution to pick from the first set)

**Output:** A set of vectors:  $W := \{\mathbf{w}_1, \dots, \mathbf{w}_{|L|}\}$ , such that

- $\mathbf{w}_i \in V_i$  for  $i = 1, \dots, |L|$  and
- $\{\mathbf{w}_1, \dots, \mathbf{w}_{|L|}\}$  forms a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{\mathbf{x}^\ell : \ell \in L\}$  for some  $L$  of maximum dimension fulfilling the  $u$ -sum condition.

##### Algorithm:

```

 $\mathbf{w}_1 := V_1[j]$  (the  $j$ 'th element on the list  $V_1$ )
for  $i \in \{2, \dots, |L|\}$  do
  for  $\mathbf{v} \in V_i$  do
    for  $\mathbf{f} \in \mathcal{W}_U$  do
      if  $\mathbf{f}(\mathbf{w}_1 + \mathbf{v}) \neq 0$  then
        ( $\mathbf{w}_1$  and  $\mathbf{v}$  are not in the same vector space)
        continue  $\mathbf{v}$ -loop (go to next  $\mathbf{v} \in V_i$ )
      end if
    end for
     $\mathbf{w}_i := \mathbf{v}$ 
  break  $\mathbf{v}$ -loop (go to next  $i \in \{2, \dots, |L|\}$ )
end for
return  $\{\mathbf{w}_1, \dots, \mathbf{w}_{|L|}\}$ 

```

---

combinations and seeing which produce the expected results in the upcoming calculations<sup>2</sup>).

In order to use this input to find the secret key,  $\mathbf{x}$ , we use the same idea as [10]. Their idea relies on the following two propositions<sup>3</sup>:

**Proposition 19.** *Let  $a \in \llbracket 0, sm - 1 \rrbracket$  and  $W = \{\mathbf{w}_1, \dots, \mathbf{w}_{|L|}\}$  be one of the sets returned by Fröbenius alignment. If  $W$  is a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{\mathbf{x}^\ell \mid \ell \in L\}$  then  $W^{p^a} \triangleq \{\mathbf{w}_1^{p^a}, \dots, \mathbf{w}_{|L|}^{p^a}\}$  is a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{\mathbf{x}^\ell \mid \ell \in p^a L\}$ , where  $p^a L \triangleq \{p^a \ell \mid \ell \in L\}$ .*

*Proof.* To ease notation, we just show the proposition for  $a = 1$ . The idea generalizes directly to other  $a$ 's.

That  $\text{span}_{\mathbb{F}_{q^m}} W^p \subseteq \text{span}_{\mathbb{F}_{q^m}} \{\mathbf{x}^\ell \mid \ell \in pL\}$  follows easily from  $z \mapsto z^p$  being  $\mathbb{F}_p$ -linear for all  $z \in \mathbb{F}_{q^m}$ . To prove that the vectors of  $W^p$  are linearly independent, we note that for  $\mathbf{w}_i \in W$ , the first entries of  $\mathbf{w}_i$  were chosen to be  $(1, \omega^i, \omega^{2i}, \dots, \omega^{|L|i})$  in section V-F, so the first entries of  $\mathbf{w}_i^p$

<sup>2</sup>The number of vector spaces of maximal dimension is not very big. The largest we saw was  $3m$ , so this step of exhaustive search is not a problem

<sup>3</sup>The authors of [10] were not able to prove the second proposition in their setting. Our proof does not apply in their setting, either, as it is not clear whether their vectors are linearly independent.

are  $(1, \omega^{ip}, \omega^{2ip}, \dots, \omega^{i|L|p})$ . Furthermore, the matrix

$$\begin{bmatrix} 1 & \omega^p & \omega^{2p} & \dots & \omega^{|L|p} \\ 1 & \omega^{2p} & \omega^{4p} & \dots & \omega^{2|L|p} \\ \vdots & & & & \vdots \\ 1 & \omega^{|L|p} & \omega^{|L|p} & \dots & \omega^{|L|^2p} \end{bmatrix}$$

has full rank, as it is a Vandermonde-matrix. Hence the vectors of  $W^p$  are linearly independent  $\square$

**Proposition 20.** For  $L_1, L_2 \subseteq \mathbb{Z}_{q^m-1}$

$$\begin{aligned} & \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_1 \} \cap \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_2 \} \\ &= \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_1 \cap L_2 \} \end{aligned}$$

*Proof.* First, note that the inclusion " $\supseteq$ " is trivial. To prove the other inclusion, we show that the two vector spaces have the same dimension. As  $\mathbf{x}^0, \dots, \mathbf{x}^{n-1}$  are linearly independent per Lemma 26, the dimension of the right hand side is  $|L_1 \cap L_2|$ . To find the dimension of the left hand side, we use the formula  $\dim(V_1 + V_2) = \dim(V_1) + \dim(V_2) - \dim(V_1 \cap V_2)$ . In our case, the sum of vector spaces is equal to

$$\begin{aligned} & \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_1 \} + \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_2 \} \\ &= \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L_1 \cup L_2 \} \end{aligned}$$

So it has dimension  $|L_1 \cup L_2|$ . From this we deduce that the left hand side has dimension

$$|L_1| + |L_2| - |L_1 \cup L_2| = |L_1 \cap L_2|$$

Exactly like the right hand side. This finishes our proof.  $\square$

Using these two ideas and the attack on GRS-codes by Sidelnikov and Shestakov [4], we can detangle the solution space for many parameters. We first illustrate the idea with an example, and then we will describe the idea in more detail.

*Example 21.* For  $q = 25, m = 2, r = 100$ , we are given  $W = \{w_1, \dots, w_{26}\}$ , which is a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L \}$  with

$$L = \llbracket 1, 20 \rrbracket \cup \{25, 30, 35, 40, 45, 50\}.$$

Using proposition 19 and 20 along with the fact that  $5^{-1} = 5^3$  in  $\mathbb{Z}_{q^m-1}$ , we see that

$$\begin{aligned} & (\text{span}_{\mathbb{F}_{q^m}} W) \cap (\text{span}_{\mathbb{F}_{q^m}} W^{5^3}) \\ &= \text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in L \cap 5^{-1}L \} \end{aligned}$$

with  $L \cap 5^{-1}L = \llbracket 1, 10 \rrbracket$ . Note that calculating a basis for  $(\text{span}_{\mathbb{F}_{q^m}} W) \cap (\text{span}_{\mathbb{F}_{q^m}} W^{5^3})$  is simple linear algebra given  $W$ . Hence we are able to find a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell : \ell \in \llbracket 1, 10 \rrbracket \}$ .

Now we just repeat the process. If we again raise to the power  $5^3$  and take the intersection, we find a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^1, \mathbf{x}^2 \}$ . This is a basis for the GRS-code  $\text{GRS}_2(\mathbf{x}, \mathbf{x})$ , so we can use the Sidelnikov-Shestakov attack [4] to find  $\mathbf{x}$ .

From this example we already see the outline of how detangle the solution sets. However, the idea used in example

21 relies on  $L$  having a very particular structure. We will call any  $L$  that has this structure well formed.

**Definition 22.**  $L \subseteq \mathbb{Z}_{q^m-1}$  is called well-formed of length  $\sigma \in \llbracket 1, sm-1 \rrbracket$  if

- $L \subseteq \llbracket 1, p^\sigma - 1 \rrbracket$
- $p^a \in L$  for all  $a \in \llbracket 0, \sigma - 1 \rrbracket$
- For every  $c \in \llbracket 2, p - 1 \rrbracket$ :

$$\forall a \in \llbracket 0, \sigma - 1 \rrbracket : p^a \cdot c \in L \Rightarrow$$

$$\forall a \in \llbracket 0, \sigma - 1 \rrbracket, \forall b \in \llbracket 2, c - 1 \rrbracket : p^a \cdot b \in L$$

Note that if such a  $\sigma$  exists, it is unique.

The remainder of this section will be used to prove that the strategy of example 21 works whenever  $L$  is well formed. In appendix B, we give experimental evidence to show that it is always possible to find an  $L$  which is well-formed when  $r$  is small enough. In section VII, we will see that the Gröbner basis step of our attack is the limiting factor, and this step has a runtime which increases in  $r$ . In the appendix we will show that it is possible to find well-formed  $L$ 's for  $r$ 's which are much bigger than the largest  $r$ 's for which we are able to run the Gröbner basis step of our attack. In other words, the problem of finding well-formed  $L$ 's is, in practice, not the limiting factor of our attack.<sup>4</sup>

To prove that the detanglement-idea of example 21 always works when  $L$  is well-formed, we start out by formalizing the step we repeated twice in the example. Here and throughout the rest of this section, we will use the notation of proposition 19:

**Definition 23.** A detanglement step is an algorithm, which takes as input  $(L, W)$ , where  $L \subseteq \mathbb{Z}_{q^m-1}$  and  $W \subseteq \mathbb{F}_{q^m}^n$ , and outputs  $(L', W')$  where  $L' = L \cap p^{-1}L$  and  $W'$  is a basis for  $\text{span}_{\mathbb{F}_{q^m}} W \cap \text{span}_{\mathbb{F}_{q^m}} W^{p^{sm-1}}$ .

Note that the output is efficiently computable from the input.

Now for the key lemma of detanglement:

**Lemma 24.** If the input  $(L, W)$  of a detanglement step satisfies

- $L$  is well formed with length  $\sigma > 1$
- $W$  is a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L \}$

Then the output  $(L', W')$  will satisfy

- $L'$  is well formed with length  $\sigma - 1$
- $W'$  is a basis for  $\text{span}_{\mathbb{F}_{q^m}} \{ \mathbf{x}^\ell \mid \ell \in L' \}$

*Proof.* As  $p^{-1} = p^{sm-1}$  in  $\mathbb{Z}_{q^m-1}$ , the statement about  $W'$  follows from proposition 19 and 20.

We now turn our attention to  $L' = L \cap p^{-1}L$ . Throughout this proof, we will use the fact that

$$b \in L' \Leftrightarrow b \in L \text{ and } pb \in L \quad (11)$$

which follows directly from the definition of  $L'$ .

<sup>4</sup>We suspect that even when no  $L$ 's are well-formed, we would still be able to detangle most solution spaces through a clever combination of the ideas in example 21 and multiple runs of a Gröbner basis algorithm with different polynomial inputs.



To show  $L' \subseteq \llbracket 1, p^{\sigma-1} - 1 \rrbracket$ , we first note that  $L' \subseteq L \subseteq \llbracket 1, p^\sigma - 1 \rrbracket$ . We claim that  $L' \cap \llbracket p^{\sigma-1}, p^\sigma - 1 \rrbracket = \emptyset$ . To prove this claim, it is sufficient to show that for every  $b \in \llbracket p^{\sigma-1}, p^\sigma - 1 \rrbracket$ ,  $pb$  is not in  $L$ . First observe that

$$b \in \llbracket p^{\sigma-1}, p^\sigma - 1 \rrbracket \Rightarrow bp \in \llbracket p^\sigma, p^{\sigma+1} - p \rrbracket$$

Next, observe that  $\llbracket p^\sigma, p^{\sigma+1} - p \rrbracket \cap L = \emptyset$  (note that here we need  $\sigma \leq sm - 1$  as it implies  $p^{\sigma+1} - p \leq q^m - 1$ , so nothing "wraps around"). These two observations together give a proof of the claim, and we can thus conclude that  $L'$  fulfills the first criteria of well formed.

For the second criteria of well formed, we have to show that  $p^a \in L'$  for all  $a \in \llbracket 0, \sigma - 2 \rrbracket$ . Per Equation (11), we just have to show that both  $p^a \in L$  and  $p \cdot p^a \in L$ , but this is clear.

For the third criteria, first note that if  $p = 2$ , the statement is vacuously true. Hence for this proof, we assume  $p > 2$ . Now let  $c \in \llbracket 2, p - 1 \rrbracket$  be given such that

$$\forall a \in \llbracket 0, \sigma - 2 \rrbracket : p^a c \in L'$$

By Equations 11 this implies

$$\forall a \in \llbracket 0, \sigma - 1 \rrbracket : p^a c \in L.$$

As  $L$  is well-formed, we can conclude that

$$\forall a \in \llbracket 0, \sigma - 1 \rrbracket, \forall b \in \llbracket 2, c - 1 \rrbracket : p^a b \in L.$$

Again by Equations 11 this implies

$$\forall a \in \llbracket 0, \sigma - 2 \rrbracket, \forall b \in \llbracket 2, c - 1 \rrbracket : p^a b \in L',$$

which is what we wanted to prove.  $\square$

If we repeat the detanglement step  $\sigma - 1$  times, we end out with an  $L$  which is well formed with length 1. I.e. we will have a basis  $W$  of  $\text{span}_{\mathbb{F}_{q^m}} \{x^\ell : \ell \in L'\}$ , where  $L' = \llbracket 1, |L'| \rrbracket$ . I.e.  $W$  is a basis for the GRS-code  $\text{GRS}_{|L'|}(x, x)$ , so we can apply the Sidelnikov-Shestakov attack [4] to find  $x$ .

The algorithm for detanglement is summarized in Algorithm V-H Detanglement.

For each  $W$ , the algorithm has to perform the detanglement step at most  $sm - 2$  times. The detanglement step is polynomial in  $n$ . Furthermore, the Sidelnikov-Shestakov attack has to be run. This also have runtime polynomial in  $n$ . As there are  $\nu$  different  $W$ 's, the total runtime of the algorithm is  $\text{poly}(n) \cdot \nu$ .

## VI. THEORETICAL ANALYSIS OF THE RUNTIME

When we analyze the runtime of the attack, it is helpful to split the attack into four parts:

- 1) Find  $L$ . The runtime of this part is not well understood, but it is potentially exponential in  $r$ .
- 2) Find a Gröbner basis in the grevlex order for the ideal generated by  $\mathcal{P}_u \cup \mathcal{L}$ . The runtime is well understood for random systems, but our results in the next section will show that our system does not behave like a random system. Hence, analyzing the precise complexity of the Gröbner basis step is a hard task. One difficulty is to predict the number of solutions to the system (see next section). In any case, even if the complexity of this step

---

### Algorithm 8 Detanglement

---

**Input:**

- $Ws = \{W_1, \dots, W_\nu\}$ , a set of bases of all vector spaces in  $\mathcal{Z}_{\mathbb{F}_{q^m}}(\mathcal{P}_u \cup \mathcal{L})$  of maximal dimension.
- $L \subseteq \mathbb{Z}_{q^m-1}$ , which is well-formed and such that  $\text{span}_{\mathbb{F}_{q^m}} W_i = \text{span}_{\mathbb{F}_{q^m}} \{x^\ell \mid \ell \in L\}$  for some  $i \in \llbracket 1, \nu \rrbracket$ .

**Output:** A vector  $v$ , that can be used as secret key to decrypt messages

```

1: for  $W \in Ws$  do
2:    $\sigma \leftarrow$  length of  $L^*$  (see Def. 22)
3:    $L' \leftarrow L$ 
4:   while  $\sigma > 1$  do
5:      $L' \leftarrow L' \cap p^{-1}L'$ 
6:      $W \leftarrow$  basis for  $(\text{span}_{\mathbb{F}_{q^m}} W) \cap (\text{span}_{\mathbb{F}_{q^m}} W^{p^{sm-1}})$ 
7:     if  $|L'| \neq |W|$  then
8:       ( $W$  is not a basis of  $\text{span}\{x^\ell \mid \ell \in L'\}$ )
9:       continue  $W$ -loop (go to next  $W \in Ws$ )
10:    end if
11:  end while
12:  Run the Sidelnikov Shestakov attack with input  $W$ 
13:   $v \leftarrow$  output of Sidelnikov Shestakov attack
14:  if  $G$  is a generator matrix for  $\mathcal{A}_r(v, v)$  then
15:    return  $v$ 
16:  end if
17: end for

```

---

remains exponential in the parameters, it is practical for some cases, and that is what we want to emphasize in this article.

- 3) Transform the Gröbner basis in the grevlex order into a Gröbner basis in the lex-order using the FGLM algorithm [17]. This algorithm has complexity  $O(ND^3)$  where  $D$  is the number of solutions to the system counted with multiplicities and  $N$  is the number of variables.  $D$  varies greatly with parameters (see next section), and therefore we cannot estimate the runtime analytically. However, once we have a Gröbner basis in the lex order, it is easy to compute the solutions of the system.
- 4) Fröbenius align and detangle the set of solutions to find one vector, which can be used like the secret key to decrypt messages. The runtime of this part is polynomial in  $n$  and  $\nu$ , the number of solutions of the system (without multiplicities).

In practice we find that part 2 and 3 are the deciding factors of whether the attack is feasible or not. Analyzing analytically their running times is a hard task, on which we are working. In the next section we have collected our initial findings on the running time of part 2.

#### A. Running time of part 2

When analyzing the runtime of the Gröbner basis algorithm, the first step is often to count the number of polynomials and the number of variables. In one way, this is easy: We have  $s \cdot k$  linear polynomials,  $k$  polynomials of degree 2 or 3 and  $n - |L|$

variables (as we can fix  $|L|$  of the  $n$  variables). However, not all of the polynomials are linearly independent.

**Proposition 25.** *The number of linearly independent polynomials in  $\mathcal{L}$  is less than or equal to  $n - |S_p|$ , where*

$$S_p := \{\ell \in \mathbb{Z}_{q^{m-1}} \mid \forall p^a \in \llbracket 1, r \rrbracket : \ell \cdot p^a \in \mathcal{C}_q\}$$

and  $\mathcal{C}_q$  is defined in section V-D.

Experimentally, we find that the inequality is an equality for all parameters we tried. To prove the proposition, we need a short lemma

**Lemma 26.** *The vectors  $\mathbf{x}^0, \dots, \mathbf{x}^{n-1}$  are linearly independent.*

*Proof.*  $\mathbf{x}$  is a vector of length  $n$ . Hence the  $n \times n$  matrix, whose columns are  $\mathbf{x}^i$  for  $i = 0, \dots, n-1$  is a square Vandermonde matrix. We know that a Vandermonde matrix has full rank iff all entries in the generating vector are different.  $\mathbf{x}$  is a permutation of  $(1, \alpha, \alpha^2, \dots, \alpha^{n-1})$ . So all entries in  $\mathbf{x}$  are different.  $\square$

Now we are ready to prove the proposition

*Proof of proposition 25.* If we define  $G_a := \sqrt[p^a]{G}$ , where the root is applied to all entries of  $G$ , then by the rank-nullity theorem, the number of linearly independent polynomials in  $\mathcal{L}$  is equal to

$$n - \dim\left(\bigcap_{p^a \in \llbracket 1, r \rrbracket} N(\sqrt[p^a]{G})\right)$$

where  $N(\sqrt[p^a]{G})$  is the null-space of  $\sqrt[p^a]{G}$ . We will now show that if  $\ell \in \mathbb{Z}_{q^{m-1}}$  satisfies  $\ell \cdot p^a \in \mathcal{C}_q$  for all  $p^a \in \llbracket 1, r \rrbracket$  then  $\mathbf{x}^\ell \in N(\sqrt[p^a]{G})$ . To do this, we first note that  $\ell \cdot p^a \in \mathcal{C}_q$  implies that

$$\sum_{j=1}^n g_{ij} x_i^{\ell \cdot p^a} = 0$$

for all  $i = 1, \dots, k$  and  $p^a \in \llbracket 1, r \rrbracket$ . But by taking the  $p^a$ 'th root, we see that this implies

$$\sum_{j=1}^n \sqrt[p^a]{g_{ij}} x_i^\ell = 0$$

for all  $i = 1, \dots, k$  and  $p^a \in \llbracket 1, r \rrbracket$ . Which is equivalent to  $\mathbf{x}^\ell \in N(\sqrt[p^a]{G})$ . Hence

$$\{\mathbf{x}^\ell \mid \forall p^a \in \llbracket 1, r \rrbracket : \ell \cdot p^a \in \mathcal{C}_q\} \subseteq \bigcap_{p^a \in U_p} N(\sqrt[p^a]{G})$$

Per Lemma 26, powers of  $\mathbf{x}$  are linearly independent, so the dimension of  $\bigcap_{p^a \in \llbracket 1, r \rrbracket} N(\sqrt[p^a]{G})$  is at least  $|S_p|$ . Hence the number of linearly independent equations is at most  $n - |S_p|$ , which finishes the proof.  $\square$

Of course, it would be nice, if we could find a closed formula for the number of linear equations, but we were unable to do this. However, we can say that

**Lemma 27.** *The size of the set in proposition 25 satisfies*

$$\lfloor \frac{r}{\max U} \rfloor \leq |S_p| \leq mr,$$

where

$$U \triangleq \{u\} \cup \{p^a \mid a \in \llbracket 0, s-1 \rrbracket, p^a \in \llbracket 1, r \rrbracket\}$$

*Proof.* Same idea as the proof of lemma 18.  $\square$

Now, if we use the linear equations of  $\mathcal{W}_U$  to eliminate variables from the equations of degree 2 or 3, we find that the total number of variables that the Gröbner basis algorithm has to deal with is

$$n - |L| - (n - |S_p|) = |S_p| - |L| \leq mr. \quad (12)$$

However, once we eliminate variables, the  $k$  equations of degree 2/3 are no longer linearly independent. Unfortunately we were not able to make much progress on how many linearly independent equations the Gröbner basis algorithm has access to once  $|L|$  variables are fixed and  $(n - |S_p|)$  are eliminated using the linear equations.

## VII. RESULTS FROM EXPERIMENTS

We split this section into two subsection. In the first, we look at the number of solutions with and without multiplicities that the Gröbner basis algorithm returns. The large variability of these numbers support our claim from the previous section that our system does not behave like a random system. In the second subsection, we report on experimental results for the runtime of the Gröbner basis algorithm.

We run magma V2.26-9 on a virtual machine with a 4x Single Core CPU running at 1996 MHz. We capped calculations at 10 GB ram, and stopped them if they ran for more than five hours.<sup>5</sup>

### A. The number of solutions to the Gröbner basis algorithm

In general each run of the Gröbner basis algorithm produces as many results as there are vector spaces of maximal dimension in  $\mathcal{Z}_{\mathbb{F}_q^m}(\mathcal{P}_u \cup \mathcal{L})$ . For most parameters, there are  $m$  vector spaces of maximal dimension, but we have also observed other possibilities. In rare instances the Gröbner Basis algorithm finds fewer solutions than there are vector spaces of maximal dimension. This can happen, as the vector spaces of maximal dimension are not disjoint<sup>6</sup>. We can also find more solutions than there are vector spaces of maximal dimension. This happens if we fix the variables in a way that allows for a solution from a vector space of lower dimension<sup>7</sup>.

<sup>5</sup>The code for our experiments are available on <https://github.com/FrejaElbro/Cryptanalysis-of-BCH-McEliece>.

<sup>6</sup>For example for  $q = 7$ ,  $m = 4$  and  $r = 11$ , both  $\{1, 2, 3, 4, 5\}$  and  $\{1, 2, 3, 4, 7\}$  fulfill the  $u$ -sum criteria. When we fix 5 variables during the Gröbner basis computation, we are essentially picking an element at random in each of the corresponding vector spaces. With probability  $1/n$  that vector also lies in the intersection of the vector spaces, so instead of one solution from each of the two vector spaces, we will find one solution, which happens to lie in both.

<sup>7</sup>See for example the sets in remark 16. With probability  $1/n$  we find a solution in the vector space  $\text{span}_{\mathbb{F}_q^m}\{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^8, \mathbf{x}^{16}\}$  when we fix 5 variables.

r	# solutions	with mult.
2	3	3
3	3	9
4	3	3
5	3	12
6	3	3
7	3	15
8	3	3
9	3	24
10	3	6 or 9
11	3	63
12	3	39
13	3	39
14	3	3
15	3	39
16	3	3
17	3	39
18	3	6
19	3	48
20	3	6
21	3	63
22	3	9
23	6	129
24	6	45
25	3	6
26	3	6
27	3	54

r	# solutions	with mult.
3	132	136
4	132	136
5	132	152
6	4	16 or 20
7	4	64
8	4	64
9	4	8 or 12
10	8	160
11	4	16
12	4	16 or 20
13	4	48

TABLE I

NUMBER OF SOLUTIONS WITH AND WITHOUT MULTIPLICITIES FOR TEN RUNS OF THE GRÖBNER BASIS ALGORITHM. ON THE LEFT FOR  $q = 13$  AND  $m = 3$  AND ON THE RIGHT FOR  $q = 4$  AND  $m = 4$

Furthermore, we sometimes find a lot of extra solutions that we have no control over. This is particularly true for  $q = 2$ . Here the runtime is dominated by part 3 in section VI. For most other parameters, part 2 of section VI is the dominating part.<sup>8</sup>

Aside from the many possibilities for the number of solutions, there is also a huge variability in the multiplicities of the solutions. The largest multiplicity we saw was for  $q = 29$ ,  $m = 2$  and  $r = 27$ . Here we found just two solutions, but each of them had multiplicity 85. A parameter set, which we though should behave fairly similarly is  $q = 29$ ,  $m = 2$  and  $r = 26$ , but here each solution only had multiplicity 5. And close to this is  $q = 29$ ,  $m = 2$  and  $r = 30$ , where each solution had multiplicity 1. We were not able to explain these large variations.

Also the multiplicities varied for different runs of the Gröbner basis algorithm. In Table I we have recorded the number of solutions with and without multiplicities for different parameter sets and ten independent runs of the Gröbner basis algorithm. For all other results than  $q = 4$ ,  $r = 3, 4, 5$ , the number of results is equal to the number of vector spaces of maximal dimension in  $\mathcal{Z}_{\mathbb{F}_q^m}(\mathcal{P} \cup \mathcal{L})$ .

### B. The runtime of the Gröbner basis algorithm

In this section, we explore how the different parameters  $r$ ,  $s$ ,  $m$  and  $p$  influence the runtime of the Gröbner basis algorithm.

<sup>8</sup>The other parameter sets where we found extra solutions that we have no control over all have small values of  $r$  and  $q = 3, 4$  or  $8$ .

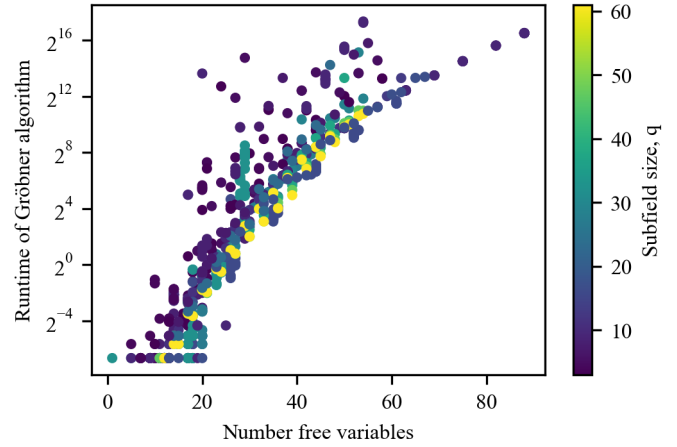


Fig. 1. The runtime of the Gröbner basis part of our attack in seconds (part 2 and 3 of section VI) as a function of number of free variables and the size of the subfield,  $q$ .

$r$  influences the runtime in two different ways. Increasing  $r$  leads both to an increase in the number of variables (Equation (12)), and a decrease in the dimension of the BCH code (Theorem 11), and hence the number of non-linear equations in  $\mathcal{P}_u$ . Predictably, the effect of increasing the number of variables is to increase the runtime of the Gröbner basis algorithm. In Fig. 1 we have plotted all our results in one graph. We see that the runtime of the Gröbner basis algorithm seems exponential in the number of variables.

In order to better be able to compare our results with information set decoding (the currently best known attack) we chose to graph our results as a function of the rate instead of  $r$  (the two are closely connected). The results are in Fig. 2. We conclude that our attack works best for high rates, which was to be expected as high rate is connected to low  $r$  and hence low number of variables and many equations. The precise cutoff point between when our attack is better than information set decoding depends on the other parameters.

The easiest parameter after  $r$  to understand is  $s$ , that is the extension degree of the base field  $\mathbb{F}_q$ . The larger the extension degree, the more linear equations can be added by taking  $p$ 'th roots, which leads to fewer variables and hence lower runtime of our attack. We ran some experiments, where we kept  $n$  constant, but varied  $s$ . Our experiment consisted of running the Gröbner basis part of our attack for larger and larger  $r$ , and stopping when either the runtime exceeded 5 hours or when memory exceeded 10 GB. The results are in Table II. We see that for both  $p = 2$  and  $p = 3$ , the effect of  $s$  is clearly to bring down runtime of the Gröbner basis algorithm, and hence bring more parameter sets into the range we can attack with our limited resources.

We now explore the effect of  $m$  on the runtime. We ran similar experiments as for  $s$  to find the results of Table III. We find that increasing  $m$  leads to us being able to break systems with higher ISD security. This is in part because increasing  $m$  increases  $n$ , which increases the expected number of rounds

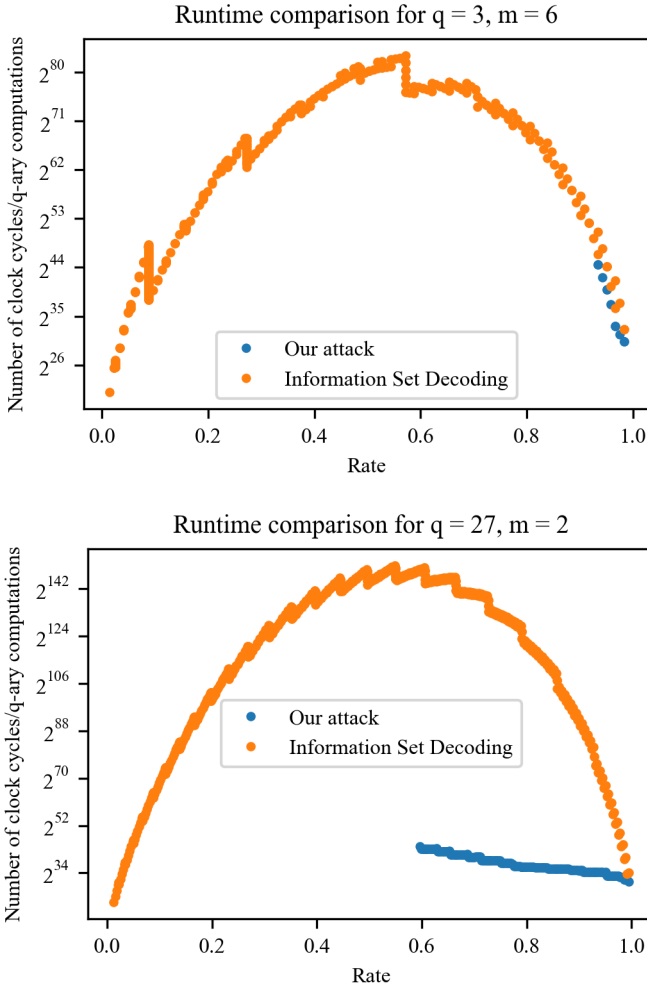


Fig. 2. Comparison between the runtime of our attack and the best previously known attack (Information set decoding). We estimate the runtime of Information Set Decoding as the expected number of  $q$ -ary computations that the Prange algorithm [2] needs to find the error-vector. For our attack, the runtime is estimated as the number of clock cycles used to run part 2 and 3 of our attack  $|L|$  times (section VI). The remaining steps of our attack are insignificant in comparison. As the rate decreases,  $r$  increases, so the number of free variables increases. This explains why the runtime of our attack increases for decreasing rates, and why we were only able to run our attack for high rates.

the Prange Algorithm [2] has to run to find the error vector. However, we can also see that when we increase  $m$ , we are able to solve systems with more variables within the same time window. We suspect this is because we are increasing  $k$ , which gives us more polynomials for the Gröbner basis algorithm to work with.

Finally, we look at the effect of  $p$ . This effect was the most difficult to isolate, as we cannot keep all the other parameters constant. Here we picked parameters so that  $n \in [600, 4000]$ . The results are in Table IV. We see that the effect of increasing  $p$  is to increase the ISD security of the largest parameters we are able to break in our setup.

TABLE II  
THE INFLUENCE OF  $s$  ON HOW ISD-SECURE SYSTEMS WE CAN BREAK

$p$	$s$	$n$	$r$	rate <sup>a</sup>	#pols <sup>b</sup>	#vars <sup>c</sup>	GB <sup>d</sup>	ISD <sup>e</sup>
2	2	4095	13	0.99	390	32	$2^{40}$	$2^{71}$
2	3	4095	35	0.97	596	39	$2^{42}$	$2^{121}$
2	4	4095	113	0.92	1317	61	$2^{47}$	$2^{247}$
3	1	728	12	0.93	398	42	$2^{44}$	$2^{51}$
3	2	728	47	0.83	426	46	$2^{49}$	$2^{87}$
3	3	728	168	0.60	365	40	$2^{43}$	$2^{150}$

<sup>a</sup>The rate of the code =  $k/n$ .

<sup>b</sup>The number of  $\mathbb{F}_{q^m}$ -linearly independent polynomials in  $\mathcal{P}_u$  once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.

<sup>c</sup>The number of variables in the system once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.

<sup>d</sup>The number of clock cycles expended on the Gröbner basis part of our attack including all  $|L|$  runs and both part 2 and 3 of section VI).

<sup>e</sup>The expected number of  $q$ 'ary operations used by Prange's algorithm (asymptotically the best generic decoding algorithm for BCH codes [3]).

TABLE III  
THE INFLUENCE OF  $m$  ON HOW ISD-SECURE SYSTEMS WE CAN BREAK

$q$	$m$	$n$	$r$	$k^a$	#pols <sup>b</sup>	#vars <sup>c</sup>	GB <sup>d</sup>	ISD <sup>e</sup>
3	5	242	12	202	170	34	$2^{45}$	$2^{38}$
3	6	728	12	680	398	42	$2^{44}$	$2^{51}$
3	7	2186	12	2130	714	50	$2^{45}$	$2^{63}$
3	8	6560	12	6496	1086	58	$2^{47}$	$2^{76}$
5	3	124	13	91	83	27	$2^{45}$	$2^{31}$
5	4	624	16	572	404	44	$2^{48}$	$2^{55}$
5	5	3124	15	3064	840	53	$2^{47}$	$2^{73}$
9	2	80	32	31	31	20	$2^{48}$	$2^{28}$
9	3	728	47	602	426	46	$2^{49}$	$2^{87}$
9	4	6560	50	6380	1686	69	$2^{49}$	$2^{168}$

<sup>a</sup>The dimension of the code. Also the number of polynomials in  $\mathcal{P}_u$ .

<sup>b</sup>The number of  $\mathbb{F}_{q^m}$ -linearly independent polynomials in  $\mathcal{P}_u$  once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.

<sup>c</sup>The number of variables in the system once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.

<sup>d</sup>The number of clock cycles expended on the Gröbner basis part of our attack including all  $|L|$  runs and both part 2 and 3 of section VI).

<sup>e</sup>The expected number of  $q$ 'ary operations used by Prange's algorithm (asymptotically the best generic decoding algorithm for BCH codes [3]).

## VIII. CONCLUSION AND FURTHER WORK

We build on the attack by [10] to develop an algebraic attack on  $q$ -ary BCH-McEliece, which demonstrates that in particular high rate codes with non-prime  $q$  are unsafe cryptographic choices.

The attack of [10] focuses on Goppa codes with large multiplicities. BCH codes in a way have the ultimate multiplicities, which motivated us to try to extend the work of [10] to BCH codes. However, through our work, we found that actually the main idea of [10], which is to throw away equations to extend the solution set, does not rely on multiplicities of the Goppa polynomial. In fact, it also applies to Classic McEliece (the NIST submission), which uses Goppa codes with irreducible



TABLE IV

THE INFLUENCE OF  $p$  ON HOW ISD-SECURE SYSTEMS WE CAN BREAK

$p$	$s$	$n$	$r$	$k^a$	#pols <sup>b</sup>	#vars <sup>c</sup>	GB <sup>d</sup>	ISD <sup>e</sup>
3	1	728	12	680	398	42	$2^{44}$	$2^{51}$
3	1	2186	12	2130	714	50	$2^{45}$	$2^{63}$
5	1	624	16	572	404	44	$2^{48}$	$2^{55}$
5	1	3124	15	3064	840	53	$2^{47}$	$2^{73}$
13	1	2196	26	2124	793	59	$2^{47}$	$2^{97}$
29	1	840	36	771	771	51	$2^{46}$	$2^{95}$
37	1	1368	34	1300	1069	51	$2^{45}$	$2^{106}$

<sup>a</sup>The dimension of the code. Also the number of polynomials in  $\mathcal{P}_u$ .<sup>b</sup>The number of  $\mathbb{F}_{q^m}$ -linearly independent polynomials in  $\mathcal{P}_u$  once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.<sup>c</sup>The number of variables in the system once the linear equations of  $\mathcal{L}$  have been used to eliminate variables and  $|L|$  variables have been fixed.<sup>d</sup>The number of clock cycles expended on the Gröbner basis part of our attack including all  $|L|$  runs and both part 2 and 3 of section VI. The remaining parts of our attack are insignificant in comparison.<sup>e</sup>The expected number of  $q$ 'ary operations used by Prange's algorithm (asymptotically the best generic decoding algorithm for BCH codes [3]).

Goppa polynomials. Breaking these codes is strongly related to finding zeros of [7]

$$\left\{ \sum_{j=1}^n g_{ij} Y_j X_j^u : i \in \{1, \dots, k\}, u \in \{0, \dots, r-1\} \right\} \quad (13)$$

I.e. here we do not have  $n$  unknowns but  $2n$  unknowns, and we cannot get linear polynomials by applying  $p$ 'th roots (we would have to apply the  $p$ 'th root to both  $\mathbf{x}$  and  $\mathbf{y}$ -variables).

However, the main idea [10] still applies. If we for example set  $U = \{0, 1, 2\}$ , then we can prove that the zero set of

$$\left\{ \sum_{j=1}^n g_{ij} Y_j X_j^u : i \in \{1, \dots, k\}, u \in U \right\},$$

contains the vector space spanned by  $\left\{ (\mathbf{y}, \mathbf{1}), (\mathbf{y}, \mathbf{x}), \dots, (\mathbf{y}, \mathbf{x}^{\lfloor \frac{r-1}{2} \rfloor}) \right\}$ . I.e. we can fix  $\lfloor \frac{r+1}{2} \rfloor$   $\mathbf{x}$ -variables, when we try to solve the system.

But even with this approach, we are not able to bring down the number of variables sufficiently for the system to be tractable by our polynomial system solver. In fact, for many parameter-sets, the reduced system has more variables than equations, even after we take advantage of the known extra solutions to fix variables. In other words, the reduced system has more solutions than we can explain.

Inspired by the work of [10], we would like to understand the additional "wrong" solutions. In [10], it is shown that even though the extra solutions of the system cannot be used directly as a secret key to decrypt messages, they can still aid in finding the secret key. It would be interesting to study whether this idea could also apply to Goppa-codes with irreducible Goppa-polynomials or BCH codes with  $q = 2$ .

## REFERENCES

[1] R. J. McEliece, *A Public-Key System Based on Algebraic Coding Theory*. Jet Propulsion Lab, 1978, pp. 114–116, dSN Progress Report 44.

[2] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962. [Online]. Available: <http://dx.doi.org/10.1109/TIT.1962.1057777>

[3] R. Canto Torres and N. Sendrier, "Analysis of information set decoding for a sub-linear error weight," in *Post-Quantum Cryptography*. Springer, 2016, pp. 144–161.

[4] V. M. Sidelnikov and S. Shestakov, "On the insecurity of cryptosystems based on generalized Reed-Solomon codes," *Discrete Math. Appl.*, vol. 1, no. 4, pp. 439–444, 1992.

[5] N. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," in *Advances in Cryptology - EUROCRYPT 2003*, ser. LNCS, vol. 2656, 2003, pp. 345–359.

[6] M. Bardet, P. Briaud, M. Bros, P. Gaborit, V. Neiger, O. Ruatta, and J. Tillich, "An algebraic attack on rank metric code-based cryptosystems," in *Advances in Cryptology - EUROCRYPT 2020 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020. Proceedings, 2020*. [Online]. Available: <http://arxiv.org/abs/1910.00810>

[7] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys," in *Advances in Cryptology - EUROCRYPT 2010*, ser. LNCS, vol. 6110, 2010, pp. 279–298.

[8] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani, "Reducing key length of the McEliece cryptosystem," in *Progress in Cryptology - AFRICACRYPT 2009*, ser. LNCS, B. Preneel, Ed., vol. 5580, Gammarth, Tunisia, Jun. 21-25 2009, pp. 77–97.

[9] R. Misoczki and P. Barreto, "Compact McEliece keys from Goppa codes," in *Selected Areas in Cryptography*, Calgary, Canada, Aug. 13-14 2009.

[10] J.-C. Faugère, L. Perret, and F. de Portzamparc, "Algebraic attack against variants of McEliece with Goppa polynomial of a special form," in *Advances in Cryptology - ASIACRYPT 2014*, ser. LNCS, vol. 8873. Kaoshiung, Taiwan, R.O.C.: Springer, Dec. 2014, pp. 21–41.

[11] W. Bosma, J. Cannon, and C. Playoust, "The Magma algebra system. I. The user language," *J. Symbolic Comput.*, vol. 24, no. 3-4, pp. 235–265, 1997, computational algebra and number theory (London, 1993). [Online]. Available: <http://dx.doi.org/10.1006/jscs.1996.0125>

[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, 5th ed. Amsterdam: North-Holland, 1986.

[13] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "Further results on Goppa codes and their applications to constructing efficient binary codes," *IEEE Trans. Inform. Theory*, vol. 22, pp. 518–526, 1976.

[14] H. Liu, C. Ding, and C. Li, "Dimensions of three types of beh codes over  $GF(q)$ ," *Discrete Mathematics*, vol. 340, no. 8, pp. 1910–1927, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0012365X1730105X>

[15] A. Dür, "The automorphism groups of Reed-Solomon codes," *Journal of Combinatorial Theory, Series A*, vol. 44, pp. 69–82, 1987.

[16] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph (algorithm 457)," *Commun. ACM*, vol. 16, no. 9, pp. 575–576, 1973.

[17] J.-C. Faugère, P. M. Gianni, D. Lazard, and T. Mora, "Efficient computation of zero-dimensional gröbner bases by change of ordering," *J. Symbolic Comput.*, vol. 16, no. 4, pp. 329–344, 1993.

## APPENDIX

## A. Overview of notation which is introduced section V

In section V we introduce a lot of additional notation. We provide the following list as an index in case our readers need to go back to look up a definition.

- $q$ , the size of  $\mathbb{F}_q$ , which is the alphabet of the BCH-code. Introduced in problem 2
- $p$ , the characteristic of the field  $\mathbb{F}_q$ . Introduced in problem 2
- $s$ , chosen such that  $p^s = q$ . Introduced in problem 2
- $m$ , the extension degree of the code. I.e. the BCH-code is a subfield subcode of a GRS-code over  $\mathbb{F}_{q^m}$ . Introduced in problem 2
- $n = q^m - 1$ , the length of the BCH-code. Introduced in problem 2

TABLE V

COMPARISON BETWEEN THE LARGEST  $r$  FOR WHICH WE ARE ABLE TO RUN THE GRÖBNER BASIS ALGORITHM (WITH THE LIMITATIONS GIVEN IN SECTION III) AND THE LARGEST  $r$  FOR WHICH WE ARE ABLE TO FIND A WELL-FORMED  $L$ .

$q$	$m$	Max $r$ for Gröbner Basis	Max $r$ with well-formed $L$
3	5	12	76
3	6	12	$\geq 157^a$
5	3	13	48
5	4	16	$\geq 213^a$
9	2	32	50
9	3	52	$\geq 303^a$
25	2	119	218
27	2	171	$\geq 424^a$

<sup>a</sup>For these parameters, it took longer than five hours to find  $L$  for larger  $r$ , so we stopped calculations.

- $r$ , the dimension of the "parent" GRS-code. Introduced in problem 2
- $k$ , the dimension of the BCH-code. Introduced in problem 2
- $G$ , the generator matrix of the BCH-code. Introduced in problem 2
- $\boldsymbol{x}$ , the secret key of the BCH-McEliece encryption scheme. Introduced in problem 2
- $\mathcal{L}$ , the linear equations in the system we will solve. Introduced in equation (5).
- $\mathcal{P}_u$ , the non-linear equations in the system we will solve. Introduced in equation (6).
- $\mathcal{C}_q$  the union of cyclotomic cosets  $\bigcup_{c \in \llbracket 1, r \rrbracket} \mathcal{C}_q(c)$ , where

$$\mathcal{C}_q(c) \triangleq \{cq^e \pmod{q^m - 1} \mid e \in \llbracket 0, m - 1 \rrbracket\},$$

Introduced in section V-D.

- $u$ -sum condition, a condition which will guarantee that a subset  $L \subseteq \mathbb{Z}_{q^m - 1}$  satisfies

$$\text{span}_{\mathbb{F}_q} \{\boldsymbol{x}^\ell \mid \ell \in L\} \subseteq \mathcal{Z}_{\mathbb{F}_q^m}(\mathcal{P}_u \cup \mathcal{L}).$$

Introduced in definition 14.

- $L$ , a subset of  $\mathbb{Z}_{q^m - 1}$  of maximal size which satisfies the  $u$ -sum condition. Introduced in section V-D
- $\nu$ , the number of subsets of  $\mathbb{Z}_{q^m - 1}$  of maximal size which satisfy the  $u$ -sum condition. Introduced in section V-F.

### B. Experimental evidence on well-formed $L$ 's

Here we provide experimental evidence to support our claim that finding well-formed  $L$ 's is not the limiting factor of our attack. More precisely, the results in table V indicates that the maximal  $r$  for which we can find a well-formed  $L$  is much bigger than the maximal  $r$  for which we can run the Gröbner basis step of our attack.

The code used to find these values and a collection of all the  $L$ 's involved can be found on <https://github.com/FrejaElbro/Cryptanalysis-of-BCH-McEliece>.