Key recovery for the McEliece cryptosystem using higher-order vanishing

Tobias Hemmert

Federal Office for Information Security, Bonn
tobias.hemmert@bsi.bund.de

Abstract. We present a new key recovery attack against the McEliece cryptosystem with binary Goppa codes that applies to a wide range of parameter sets. Earlier work already observed that homogeneous polynomials vanishing of higher order at the columns of the parity check matrix of a given code could be used to distinguish binary Goppa codes from random linear codes. This work now exploits the structure of these polynomials for key recovery. We illustrate the effectiveness of our approach by breaking a number of previously unbroken McEliece key recovery challenges published by the Technology Innovation Institute (TII). We conjecture that our approach also applies to Classic McEliece parameter sets and yields a key recovery algorithm with roughly the same complexity as the higher-order vanishing distinguisher. While this complexity is still significantly higher than the targeted security level of Classic McEliece parameter sets, this improves on currently known key recovery attacks.

1 Introduction

The McEliece encryption scheme, originally proposed in [16], is one of the oldest public-key cryptosystems. Classic McEliece, a key encapsulation mechanism that was a candidate in the fourth round of the NIST standardisation process for post-quantum cryptography, is based on the original proposal. It was not selected by NIST for standardisation mostly due to its large public key size [1]. However, it has recently been standardised at ISO/IEC [14].

At the heart of the (Classic) McEliece scheme are binary Goppa codes, a family of error-correcting codes for which efficient decoding algorithms are known. A secret key for the McEliece scheme essentially consists of a 'nice' basis of a randomly selected binary Goppa code that allows for efficient error decoding. The public key essentially comprises a *random* basis of the binary Goppa code. The security of the McEliece scheme relies on the assumption that given only such a random basis, it is not possible to efficiently decode errors for this code. In fact, for most parameter sets, including Classic McEliece parameters, the best known attack is to treat the given code as a general linear code and to apply generic decoding algorithms known as information-set decoding. The parameters for Classic McEliece have been chosen such that the complexity of decoding using the best known information-set decoding algorithms is higher than the targeted security levels.

Key recovery for the McEliece cryptosystem entails recovering the 'nice' basis of the binary Goppa code from a random basis, in other words recovering the secret Goppa code structure. However, for most parameter sets, the cost of such a key recovery attack is estimated to be much higher than the cost of decoding using generic information-set decoding algorithms. See below for an overview of what is known about key recovery attacks against the McEliece cryptosystem.

An approach that has proven to be fruitful to try to make progress on the key recovery problem for the McEliece cryptosystem has been the study of a closely related but potentially easier problem. It is natural to ask: Given a random basis (a) of a randomly selected (binary) Goppa code, or (b) of a randomly selected linear code of a fixed dimension, can one determine if one has been given (a) or (b)? We will call this the *distinguishing problem for Goppa codes*. If this distinguishing problem is intractable, then the security of Classic McEliece reduces to the assumption that the decoding problem for general linear codes is intractable. For all we know, this is indeed the case since this problem is known to be NP-complete [4] and also believed to be hard on average. Conversely, an efficient solution for the distinguishing problem need not necessarily imply a solution to the key recovery problem for Classic McEliece, but it may be an indication of structural properties that could be exploited for such an attack.

Previous work on the distinguishing problem. For binary Goppa codes as well as the more general class of alternant codes, over the years, progress has been made on the problem of distinguishing these codes from random linear codes. [11] introduces a polynomial-time distinguisher for very high-rate alternant or Goppa codes and [19] reinterpretes it as computing the dimension of the Schur square of the dual code. In what follows, we will call this the *square distinguisher*. [8] introduces the Pfaffian distinguisher which works by detecting low-rank quadratic forms that vanish when being evaluated at the columns of a parity check matrix of a Goppa or alternant code. Unlike the square distinguisher, it can be applied to a wider set of code parameters including Classic McEliece parameter sets. Its complexity is lower than the complexity of previously known key recovery attacks, but grows rapidly outside the high-rate regime and is much higher than the targeted security levels of Classic McEliece parameter sets. The syzygy distinguisher [20] uses Betti numbers of ideals in order to distinguish Goppa or alternant codes from random linear codes and yields the first such algorithm which is subexponential in the error correcting capability of the code. The higher-order vanishing distinguisher [13] computes homogeneous polynomials that vanish of higher order when evaluated at the columns of a parity check matrix as a distinguisher. It is currently estimated to be the most efficient distinguisher in particular for Goppa codes with parameters as in Classic McEliece, but its complexity is still significantly higher than their targeted security levels.

Previous work on key recovery for the McEliece cryptosystem. Over the years, variants of the McEliece cryptosystem have been proposed that replace binary Goppa codes by some other family of efficiently decodable error-correcting codes. The main motivation is to reduce the rather large public key size. It has turned

out that most of these proposals admit an efficient key recovery attack, see for example [22][24][7][12]. Besides binary Goppa codes, a notable exception so far is the use of quasi-cyclic moderate-density parity check codes that are used in the BIKE key encapsulation mechanism, which was a candidate in the fourth round of the NIST standardisation process for post-quantum cryptography.

Some progress on the key recovery problem has been made for general alternant codes and for Goppa codes over large fields. [9] describes a key recovery attack for wild Goppa codes over quadratic extensions. [3] introduces a key recovery attack for high-rate alternant codes in the same parameter regime where the square distinguisher applies. It succeeds for general high-rate alternant codes with high probability, but fails in the special case of Goppa codes. The same paper that introduces the Pfaffian distinguisher [8] also contains an efficient key recovery attack that works for high-rate alternant codes over large fields.

However, the key recovery problem for the original McEliece scheme based on binary Goppa codes has proven to be very difficult. Until recently, for binary Goppa codes (and in particular for those parameter regimes used in Classic McEliece), the best known algorithm has been to search for the Goppa polynomial using brute force and to then recover the support using the support-splitting algorithm [21]. However, this algorithm is vastly more costly than the application of information-set decoding algorithms. For example, for the smallest Classic McEliece parameter set, [8] estimates a complexity of 2^{3238} for key recovery using this approach.

For binary Goppa codes of degree 2, [17] describes an attack and uses it to break some TII challenges. For very high-rate Goppa codes, the recent publication [2] describes a key recovery attack with polynomial running time. It builds on ideas from the Pfaffian distinguisher [8] and breaks some TII challenges in practice. Very recently, [5] introduced a new key recovery attack that is conjectured to also apply to Classic McEliece parameter sets. For these, the estimated complexity is still significantly higher than the targeted security level, but much lower than the brute force attack using the support-splitting algorithm.

Our contribution. We introduce a new key recovery attack for the McEliece cryptosystem that is based on the higher-order vanishing distinguisher [13]. It also draws on ideas from previous works, in particular [2] and [8], and has the following features:

- Our algorithm applies to a wide range of binary Goppa code parameters, namely to most of the binary Goppa codes which are distinguishable from general linear codes using the higher-order vanishing distinguisher. Interestingly, for binary Goppa codes, it does not apply in a straightforward manner in distinguishability degree $p = 2$ where the higher-order vanishing distinguisher is equivalent to the square distinguisher, see Remark 3. Note that most codes that are distinguishable in degree $p = 2$ are also distinguishable in degree $p = 3$, so by passing to $p = 3$, we can usually still apply our key recovery attack.

Table 1. TII challenges broken by our approach. n denotes the length, m the field extension degree and r the degree of the underlying Goppa code. The security level λ_{TII} is tabulated as estimated by the TII. The parameter p is the degree of the higher-order vanishing polynomials we compute and s_* is the number of positions in which we shorten the dual of the Goppa code. We also tabulate the running time (wall time) of our SageMath implementation as well as the required memory. We used a machine equipped with two AMD EPYC 9745 128-Core Processors.

n	m	r	λ_{TII}	p	s_*	Wall time	Memory	Previous attack
122	7	5	86 bits	4	5	3min	0.7GB	-
121	7	5	90 bits	4	5	3min	0.7GB	-
120	7	5	94 bits	4	5	4min	0.7GB	-
253	8	5	83 bits	4	5	13min	1.7GB	[18], [2], [5]
252	8	5	89 bits	4	5	9min	1.7GB	[2], [5]
254	8	6	84 bits	3	13	16min	0.3GB	-
253	8	6	91 bits	3	13	21min	0.3GB	[18]
254	8	7	92 bits	4	12	1:46h	9.6GB	-
251	8	7	110 bits	4	12	2:16h	9.6GB	-
240	8	7	160 bits	4	12	1:10h	9.6GB	-
237	8	7	171 bits	4	12	1:26h	9.6GB	-
235	8	7	178 bits	4	12	2:33h	9.6GB	-
224	8	7	212 bits	4	12	1:57h	9.6GB	-
221	8	7	220 bits	4	12	1:21h	9.6GB	-
256	8	8	85 bits	4	16	2:46h	19.3GB	-
255	8	8	93 bits	4	16	5:05h	19.3GB	-
254	8	8	100 bits	4	16	3:51h	19.5GB	-
508	9	7	119 bits	3	17	14:29h	0.5GB	[5]
507	9	7	125 bits	3	17	8:09h	0.5GB	[2]
500	9	7	166 bits	3	17	14:31h	0.5GB	[2], [5]
491	9	7	210 bits	3	17	23:39h	0.5GB	[2], [5]
483	9	7	244 bits	3	17	7:21h	0.5GB	-
482	9	7	248 bits	3	17	17:18h	0.5GB	[5]
512	9	8	96 bits	3	21	14:23h	0.6GB	-
511	9	8	105 bits	3	21	13:24h	0.6GB	-
510	9	8	113 bits	3	21	23:58h	0.6GB	-
507	9	8	134 bits	3	21	13:10h	0.6GB	-
502	9	8	164 bits	3	21	2:18h	0.6GB	-
495	9	8	200 bits	3	21	14:41h	0.6GB	-
499	9	9	189 bits	4	20	27:15h	211.6GB	-
495	9	9	209 bits	4	20	33:32h	211.6GB	-
492	9	9	223 bits	4	20	47:45h	211.6GB	-

– We demonstrate the effectiveness of our algorithm in practice by breaking a significant number of previously unbroken TII McEliece key recovery challenges[1] with diverse parameter sets in practice. An overview of the challenges we have broken so far can be found in Table 1. The table also indicates which of these challenges have been reported broken before. The secret keys that we obtained as well as a SageMath script to verify their correctness can be found on https://github.com/tobhem/key-recovery-mceliece-tii-solutions. Note that [18] basically uses the brute force approach using an efficient implementation of the support-splitting algorithm. [18, Table 2] suggests that some more challenges with security level < 100 bits (as estimated by the TII) could be within reach of that approach, even though a practical break was not reported in that paper.
– We conjecture that our attack also applies to Classic McEliece parameters and estimate that it has roughly the same complexity as the higher-order vanishing distinguisher for those parameter sets. For the smallest Classic McEliece parameter set, this boils down to an estimated complexity of roughly 2^{300}. While this is of course still higher than the targeted security level of 128 bits, it significantly improves on previously known key recovery attacks.
– For simplicity and since it is the cryptographically most interesting case, we phrase all results and in particular the key recovery algorithm to apply to binary Goppa codes. However, we have verified that, with minor adaptions, our approach can be applied more generally to recover the support and multiplier of alternant codes in arbitrary characteristic.

Our approach. We sketch some core ideas of our key recovery algorithm by illustrating it with a binary alternant code which is square-distinguishable, i.e. where the higher-order vanishing distinguisher works in degree $p = 2$. We remarked above that our key recovery algorithm does not work for binary *Goppa* codes in degree $p = 2$. However, for random alternant codes, it turns out the complications that prevent an application of our attack in degree $p = 2$ for Goppa codes do not occur.

Let $\boldsymbol{x} \in \mathbb{F}_{2^9}^{360}$ have pairwise distinct entries and $\boldsymbol{y} \in \mathbb{F}_{2^9}^{360}$ have non-zero entries. Let $C = \mathrm{Alt}_3(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_2$ of degree $r = 3$, extension degree $m = 9$ and length $n = 360$. See Definition 3 for a definition of alternant codes. By Proposition 2, we know that

$$\mathrm{span}_{\mathbb{F}_{512}}(C^{\perp}) = \sum_{i=0}^{8} \mathrm{GRS}_3(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

where, for a subspace $U \subset \mathbb{F}_{2^m}^n$, the subspace $U^{(2)}$ comprises all vectors that are the componentwise Frobenius automorphism applied to all vectors in U. Our

[1] See https://www.herox.com/TIIMcElieceChallenges

aim is to retrieve a basis of the subspace

$$W_0 \coloneqq \sum_{i=0}^{7} \mathrm{GRS}_3(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

that misses one of the GRS blocks. If we can do this, then we can compute a basis of

$$W_0 \cap W_0^{(2)} \cap \ldots \cap W_0^{(2^7)} = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^7)}$$

This is a GRS code, and we can then retrieve a support and multiplier using the Shestakov-Sidelnikov attack [22], which is then also a valid support and multiplier for the alternant code.

Thus, we are left with the problem of computing a basis of W_0. This is where we use the higher-order vanishing distinguisher. First, note that a generator matrix of $\mathrm{span}(C^\perp)$ is given by

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}} \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2)} \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^2)} \\ \cdots \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^8)} \end{pmatrix} \text{ with } \boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_1x_1 & y_2x_2 & \ldots & y_nx_n \\ y_1x_1^2 & y_2x_2^2 & \ldots & y_nx_n^2 \end{pmatrix}$$

We denote the column vectors of $\boldsymbol{A}$ by $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{F}_{512}^{mr}$.

The higher-order vanishing distinguisher in degree $p = 2$, which is equivalent to the square distinguisher, now considers the linear map

$$\varphi_{\boldsymbol{A}}^{(2)} \colon \mathbb{F}_{512}\left[X_i^{(j)} \mid 0 \le i \le 2, 0 \le j < 9\right]^{(2)} \to \mathbb{F}_{512}^n, \quad f \mapsto (f(\boldsymbol{a}_1), \ldots, f(\boldsymbol{a}_n))$$

where the domain consists of the vector space of homogeneous quadratic polynomials in the indicated indeterminates. The map evaluates a given homogeneous quadratic polynomial at all columns of $\boldsymbol{A}$. Note that for simplicity, we indexed the mr indeterminates $X_i^{(j)}$ for which we substitute the entries in a given column of $\boldsymbol{A}$ such that j corresponds to the block $(\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^j)}$ and i corresponds to the ith entry in that block.

The distinguisher exploits that due to the structure of $\boldsymbol{A}$, there are non-trivial elements in the kernel of $\varphi_{\boldsymbol{A}}^{(2)}$. For example:

- $X_0^{(0)}X_2^{(0)} - X_1^{(0)}X_1^{(0)}$ is in the kernel because $y_i \cdot y_i x_i^2 = y_i x_i \cdot y_i x_i$ for all $1 \le i \le n$
- More generally, all polynomials $X_0^{(j)}X_2^{(j)} - X_1^{(j)}X_1^{(j)}$ for $0 \le j < m$ are in the kernel.
- $X_2^{(0)}X_0^{(1)} - X_0^{(0)}X_1^{(1)}$ is in the kernel because $y_i x_i^2 \cdot (y_i)^2 = y_i \cdot (y_i x_i)^2$ for all $1 \le i \le n$
- More generally, all polynomials $X_2^{(j)}X_0^{(j+1)} - X_0^{(j)}X_1^{(j+1)}$ for $0 \le j < m$ are in the kernel.

However, note that one cannot construct kernel elements of the form $X_{i_1}^{(0)} X_{i_2}^{(2)} - X_{i_3}^{(0)} X_{i_4}^{(2)}$. The reason is that $r = 3$ and we would need $r \geq 5$ in order to have identities such as $y_i \cdot (y_i x_i)^{2^2} = y_i x_i^4 \cdot (y_i)^{2^2}$ translate into polynomials in the $X_i^{(j)}$ that lie in the kernel. Extensive experiments indeed confirm that one can only 'mix' indeterminates from two adjacent blocks to obtain kernel elements, and that these yield a complete description of the kernel.

[13, Proposition 8] already observed that the code with generator matrix $\boldsymbol{A}$ could be shortened in a certain number of positions while still maintaining a non-trivial kernel. Once a certain value of shortened positions $s_{\max}$ is exceeded, the kernel becomes trivial. In our concrete example above, we have $s_{\max} = 1$. One observation at the heart of our key recovery attack is the following:

Heuristic 1. Let $\boldsymbol{A}_{s_{\max}}$ be a generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^{\perp})$ shortened in $s_{\max} = 1$ positions. The kernel of $\varphi^{(2)}_{\boldsymbol{A}_{s_{\max}}}$ has dimension $m = 9$ (i.e. the field extension degree). Furthermore, it is spanned by certain polynomials $h_0, \ldots, h_{m-1}$ that, up to a linear transformation of variables, each only depends on indeterminates $X_i^{(j)}$ from two adjacent blocks (in the sense described above).

Suppose we have indexed the h_j such that, up to a change of variables, each h_j only mixes indeterminates from blocks j and $j+1$. Note that then any linear combination of four of the h_j must miss indeterminates from at least one of the $m = 9$ GRS blocks! It is not hard to see (this is basically the content of Lemma 5) that given one such polynomial, one can derive vectors in the subspace W_0, even without knowing in advance the linear transformation of variables that maps h_j to a polynomial that depends only on indeterminates $X_i^{(j)}$ and $X_i^{(j+1)}$ for $0 \leq i < r$.

One complication is, of course, that we are not given a 'nice' generator matrix $\boldsymbol{A}$ as above, but some random row-equivalent matrix $\boldsymbol{A}'$. We can still compute a basis of elements $g_0, \ldots, g_{m-1}$ that span the kernel of $\varphi^{(2)}_{\boldsymbol{A}'_{s_{\max}}}$. However, they need not agree with the h_j which are characterised by the fact that, after a linear change of variables, they only depend on a small number of indeterminates. In section 4, we present a criterion that detects whether, up to a linear change of variables, a given polynomial depends on at most B indeterminates, where B is a bound that can be freely chosen. This allows us to run a brute-force search through all linear combinations of the g_j until we obtain some polynomial g that depends on at most $B = 16$ variables. Since any linear combination of four of the h_j depends on at most $4 \cdot 4 = 16$ indeterminates, it is reasonable to hope that g is such an element. Indeed, this turns out to be the case with high probability. As indicated above, it is then possible to compute vectors in the subspace W_0 from g.

We can repeat this process by shortening in a different position until we have obtained a basis of W_0, and then we can finish the key recovery as explained before.

The above sketch of our key recovery algorithm is very rough and we have omitted many technical details for which we refer the reader to the rest of the

paper. We only remark that in essence, all the observations and considerations can be generalised to the situation of binary Goppa codes that are distinguishable with the higher-order vanishing distinguisher in some degree $p \geq 3$. An essential ingredient is a proof in section 3 of how much we can shorten the dual of a binary Goppa code while maintaining a non-trivial kernel of $\varphi^{(p)}$. It improves on [13, Proposition 9] and in all our experiments, it turned out that the expression we give is indeed the maximal shortening we can obtain while maintaining a non-trivial kernel of $\varphi^{(p)}$. With this amount of shortening, an appropriately adapted version of Heuristic 1 then still holds. Note however that our approach does not work for $p = 2$ in the case of binary Goppa codes because an analogue of Heuristic 1 does not hold for $p = 2$. We make some remarks on this in Remark 3.

Structure of the paper. In section 2, we recall basic definitions and properties of Goppa codes and introduce the key recovery problem for the McEliece cryptosystem as well as some basic algorithms related to it. In section 3, we recall the higher-order vanishing distinguisher from [13]. Furthermore, we prove an improved result stating in how many positions the dual of a binary Goppa code can be shortened such that there are still non-trivial higher-order vanishing polynomials for it. This is the most technical part of the paper and crucial for our key recovery attack. In section 4, the notion of essential variables of polynomials is explained and basic properties are proved. These results are then applied to higher-order vanishing polynomials. This is crucial to be able to compute higher-order vanishing polynomials that, after a suitable linear change of variables, depend on few indeterminates and that allow for key recovery. Finally, section 5 presents the key recovery algorithm, and section 6 applies it to TII challenges and makes an estimate of the complexity for Classic McEliece parameter sets.

Notation. We denote vectors by bold lower-case letters and matrices by bold upper-case letters. Given a vector subspace $U \subset \mathbb{F}_{2^m}^n$, we denote by

$$U^{(2^i)} \coloneqq \{(u_1^{2^i}, \ldots, u_n^{2^i}) \mid (u_1, \ldots, u_n) \in U\}$$

the space obtained by coordinate-wise application of the ith power of the Frobenius endomorphism to all vectors in U.

For a polynomial ring $S = \mathbb{F}[X_1, \ldots, X_k]$ over a field $\mathbb{F}$, we denote by $\mathbb{F}[X_1, \ldots, X_k]^{(p)}$ the vector subspace of all homogeneous polynomials in S of degree p.

Acknowledgements. I would like to thank Andreas Wiemers for interesting and helpful discussions. Essential ideas in this paper stem from our joint work [13].

2 Goppa codes and the McEliece cryptosystem

This section contains some generally well-known background material on Goppa codes and the McEliece cryptosystem. In subsection 2.1, we recall the definition of Goppa codes as well as their basic properties that we will need in this paper.

Goppa codes are defined by a support vector and a vector of multipliers that are derived from a Goppa polynomial. In subsection 2.2, we consider the question in what sense these are (not) unique. In subsection 2.3, we introduce the key recovery problem for the McEliece cryptosystem and explain in particular how the Goppa polynomial can be recovered given only the secret support vector of the underlying Goppa code.

2.1 Definition and basic properties of Goppa codes

Let q be a prime power. Recall that a linear $[n, k]$-code over $\mathbb{F}_q$ is a k-dimensional subspace $C \subset \mathbb{F}_q^n$. In this work, we are mainly interested in (binary) Goppa codes and more generally alternant codes, which are closely related to generalised Reed-Solomon codes. For convenience, we recall their definitions.

Definition 1. *Let q be a prime power and suppose $x_i, y_i \in \mathbb{F}_q$ for $1 \leq i \leq n$ such that the x_i are pairwise distinct and $y_i \neq 0$ for all i. The* generalised Reed-Solomon (GRS) code *of dimension r is defined by*

$$\mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) := \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbb{F}_q[X],\ \deg(f) < r\}$$

Alternant codes are defined by taking a GRS code over a larger field $\mathbb{F}_{q^m}$ and restricting to those vectors whose entries all lie in the subfield $\mathbb{F}_q$. Thus, an alternant code is a subfield subcode of a GRS code:

Definition 2. *Let q be a prime power and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ have distinct entries and $\boldsymbol{y} \in (\mathbb{F}_{q^m}^*)^n$. The* alternant code *with* support $\boldsymbol{x}$ *and* multiplier $\boldsymbol{y}$ *and* degree r *over $\mathbb{F}_q$ is*

$$\mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n$$

Such an alternant code is proper *if it has dimension $n - mr$.*

Note that in the definition, we define alternant codes to be the subfield subcode of the *dual* of a GRS code. However, this is mere convention since duals of GRS codes are again GRS codes with the same support but a different multiplier [15, Ch. 10, Theorem 4].

Goppa codes are alternant codes with a special choice of multiplier:

Definition 3. *Let q be a prime power and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ have distinct entries and $g \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree t with $g(x_i) \neq 0$ for all $1 \leq i \leq n$. The* Goppa code *with* support $\boldsymbol{x}$ *and* Goppa polynomial g *is defined as*

$$\mathrm{Gop}(\boldsymbol{x}, g) := \mathrm{Alt}_t(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{y} := \frac{1}{g(\boldsymbol{x})} := \left(\frac{1}{g(x_1)}, \ldots, \frac{1}{g(x_n)}\right)$. It is called proper *if it has dimension $n - mr$, and* irreducible *if g is an irreducible polynomial.*

The following proposition provides an alternative characterisation of Goppa codes. A proof can be found in [15, Ch. 12.3 Theorem 4].

Proposition 1. *Let q be a prime power and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ have distinct entries and $g \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree t with $g(x_i) \neq 0$ for all $1 \leq i \leq n$. Then*

$$\mathrm{Gop}(\boldsymbol{x}, g) = \left\{ (a_1, \ldots, a_n) \in \mathbb{F}_q^n \mid g(X) \text{ divides } \sum_{i=1}^{n} a_i \prod_{j \neq i} (X - x_j) \right\}$$

It is well-known [22] that an instantiation of the McEliece cryptosystem with GRS codes (instead of binary Goppa codes) allows for an efficient key recovery attack. The following proposition shows that if we take the dual of an alternant code and extend scalars to the extension field $\mathbb{F}_{q^m}$, the resulting code decomposes as a sum of GRS codes. This is a crucial property that we will use to reduce key recovery for McEliece with binary Goppa codes to key recovery for GRS codes.

Proposition 2. *Let q be a prime power and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ have distinct entries, $\boldsymbol{y} \in (\mathbb{F}_{q^m}^*)^n$, and $\mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_q$. Then*

$$\mathrm{span}_{\mathbb{F}_{q^m}} \left(\mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} \right) = \sum_{i=0}^{m-1} \mathrm{GRS}_r \left(\boldsymbol{x}, \boldsymbol{y} \right)^{(q^i)}$$

Proof. See [3, Proposition 14]. □

We will mostly be interested in *binary* Goppa codes, i.e. those with $q = 2$. A proof of the following lemma can be found in [13, Lemma 1].

Lemma 1. *Suppose $g \in \mathbb{F}_{2^m}[X]$ is square-free of degree $t \leq \frac{n}{2}$ and let $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ such that the entries x_i are pairwise distinct and $g(x_i) \neq 0$ for all $1 \leq i \leq n$. Let $\boldsymbol{y} = g(\boldsymbol{x})^{-1} \in \mathbb{F}_{2^m}^n$. Then*

$$\mathrm{GRS}_t(\boldsymbol{x}, \boldsymbol{y}) + \mathrm{GRS}_t(\boldsymbol{x}^2, \boldsymbol{y}^2) = \mathrm{GRS}_{2t}(\boldsymbol{x}, \boldsymbol{y}^2)$$

As a direct consequence of Lemma 1 and Proposition 2, we deduce:

Corollary 1. *Suppose $g \in \mathbb{F}_{2^m}[X]$ is square-free of degree $t \leq \frac{n}{2}$ and let $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ such that the entries x_i are pairwise distinct and $g(x_i) \neq 0$ for all $1 \leq i \leq n$. Then $\mathrm{Gop}(\boldsymbol{x}, g) = \mathrm{Gop}(\boldsymbol{x}, g^2)$.*

2.2 (Non-)Uniqueness of support and multiplier

Given a GRS code $C = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$, the choice of support $\boldsymbol{x} \in \mathbb{F}_q^n$ and multiplier $\boldsymbol{y} \in \mathbb{F}_q^n$ is clearly not unique. For example:

- For any $\lambda \in \mathbb{F}_q^*$, we have $\mathrm{GRS}_r(\boldsymbol{x}, \lambda \boldsymbol{y}) = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$.
- Given an affine map $\gamma \colon \mathbb{F}_q^n \to \mathbb{F}_q^n, \quad \boldsymbol{z} \mapsto a\boldsymbol{z} + b$ where $a, b \in \mathbb{F}_q$ and $a \neq 0$, we have $\mathrm{GRS}_r(\gamma(\boldsymbol{x}), \boldsymbol{y}) = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$.

However, in general, these transformations do not yield all possible supports and multipliers for C. [10] introduces Cauchy codes, which can be viewed as generalisations of GRS codes to the projective line $\mathbb{P}^1(\mathbb{F}_q^n) = \mathbb{F}_q^n \cup \{\infty\}$, and characterises all supports and multipliers that yield the same Cauchy code. To this end, for given $\lambda \in \mathbb{F}_q^*$ and $a, b, c, d \in \mathbb{F}_q$ with $ad - bc \neq 0$, we define the map

$$f\colon \mathbb{F}_q^n \times \left(\mathbb{F}_q^*\right)^n \to \mathbb{P}^1(\mathbb{F}_q^n) \times \left(\mathbb{F}_q^*\right)^n, (\boldsymbol{x}, \boldsymbol{y}) \mapsto \left(\frac{a\boldsymbol{x}+b}{c\boldsymbol{x}+d}, \lambda \cdot (c\boldsymbol{x}+d)^{r-1}\boldsymbol{y}\right) \quad (1)$$

Note that the special case $c = 0$ yields the affine transformations that we just considered. The following proposition shows that given one support-multiplier pair of a GRS code, one can obtain all others by applying functions of the above form:

Proposition 3. *Suppose $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{F}_q^n$ each have pairwise distinct entries and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{F}_q$ such that $y_i, y_i' \neq 0$ for all $1 \leq i \leq n$. Let $2 \leq r \leq n-2$. Then $\mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{GRS}_r(\boldsymbol{x}', \boldsymbol{y}')$ if and only if $(\boldsymbol{x}', \boldsymbol{y}') = f(\boldsymbol{x}, \boldsymbol{y})$ for some f as in Equation 1 with $cx_i + d \neq 0$ for all $1 \leq i \leq n$.*

Proof. This follows immediately from [10, Theorem 4]. □

For alternant codes, we deduce:

Proposition 4. *Suppose $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{F}_{q^m}^n$ each have pairwise distinct entries and $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{F}_{q^m}$ such that $y_i, y_i' \neq 0$ for all $1 \leq i \leq n$. We denote by $\phi\colon \mathbb{F}_{q^m}^N \to \mathbb{F}_{q^m}^N, \boldsymbol{z} \mapsto \boldsymbol{z}^q$ the coordinate-wise Frobenius endomorphism. Suppose*

$$(\boldsymbol{x}', \boldsymbol{y}') = \phi^i(f(\boldsymbol{x}, \boldsymbol{y})) \quad (2)$$

for some $0 \leq i < m$ and f as in Equation 1 with $cx_i + d \neq 0$ for all $1 \leq i \leq n$. Then $\mathrm{Alt}_r(\boldsymbol{x}', \boldsymbol{y}') = \mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})$.

Proof. This follows directly from Proposition 3 and the fact that the Frobenius automorphism fixes the subfield $\mathbb{F}_q$. □

Remark 1. Suppose that $C = \mathrm{Gop}(\boldsymbol{x}, g) = \mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})$ is a Goppa code, i.e. that $\boldsymbol{y} = \frac{1}{g(\boldsymbol{x})}$ where $g \in \mathbb{F}_{q^m}[X]$ is of degree r. Then $\mathrm{Alt}_r(\boldsymbol{x}', \boldsymbol{y}') = C$ holds for any $(\boldsymbol{x}', \boldsymbol{y}')$ obtained as in Equation 2. However, there need not necessarily exist a polynomial $\tilde{g}$ of degree r such that $\boldsymbol{y}' = \frac{1}{\tilde{g}(\boldsymbol{x}')}$ for the new support-multiplier pair. In other words, not all support-multiplier pairs that define a Goppa code *as an alternant code* need to originate from a Goppa polynomial. Indeed, [17] shows that this holds only if f is an affine transformation, see the following proposition.

Proposition 5. *Let $\mathrm{Gop}(\boldsymbol{x}, g) = \mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})$ be a proper Goppa code with $\boldsymbol{y} = \frac{1}{g(\boldsymbol{x})}$ and $\deg(g) = r$. Let $(\boldsymbol{x}', \boldsymbol{y}') = \phi^i(f(\boldsymbol{x}, \boldsymbol{y}))$ for a transformation f as in (1) and the coordinate-wise Frobenius endomorphism $\phi\colon \mathbb{F}_{q^m}^N \to \mathbb{F}_{q^m}^N, \boldsymbol{z} \mapsto \boldsymbol{z}^q$ and some $0 \leq i < m$. Then $\boldsymbol{y}' = \frac{1}{\tilde{g}(\boldsymbol{x})}$ for some polynomial $\tilde{g}$ with $\deg(\tilde{g}) = r$ if and only if f is affine, i.e. $c = 0$.*

Algorithm 1 Key generation algorithm for the McEliece cryptosystem

Input: $r, m, n \in \mathbb{N}$ with $n \leq 2^m$ and $mr < n$
Output: $\text{pk} \in \mathbb{F}_2^{mr \times n}$ and $\text{sk} \in \mathbb{F}_2^n \times \mathbb{F}_{2^m}[X]$

1: Randomly sample distinct $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{F}_{2^m}$.
2: Randomly sample an irreducible monic $g \in \mathbb{F}_{2^m}[X]$ with $\deg(g) = r$.
3: Compute the parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{mr \times n}$ of $\text{Gop}(\boldsymbol{x}, g)$ in reduced row-echelon form.
4: If $\text{rk}(\boldsymbol{H}) \neq mr$, return to Step 1.
5: **return** $\text{pk} = \boldsymbol{H}$ and $\text{sk} = (\boldsymbol{x}, g)$.

Proof. It is straightforward to check that the application of the Frobenius endomorphism preserves the property that a given multiplier vector is linked to a given support vector via some Goppa polynomial of degree r. For the statement about f, see [17, Proposition 3.3]. □

2.3 The McEliece cryptosystem and the key recovery problem

Algorithm 1 describes a simplified version of the Classic McEliece key generation algorithm that suffices for our purposes. It randomly selects a proper irreducible binary Goppa code with specified parameters. The support and Goppa polynomial allow the owner of the secret key to efficiently decode errors for the selected Goppa code, while the public key consists of an echelonised parity check matrix $\boldsymbol{H}$ of the Goppa code. The IND-CCA security of Classic McEliece relies on the assumption that given only $\boldsymbol{H}$, decoding random errors for the code with parity check matrix $\boldsymbol{H}$ is hard.

The best known attack against this assumption is the application of generic information-set decoding algorithms to the code with parity check matrix $\boldsymbol{H}$. This essentially treats the underlying code as a general linear code without using the fact it is actually a binary Goppa code. Of course, recovering a valid support and Goppa polynomial for the code with parity check matrix $\boldsymbol{H}$ also allows one to efficiently decode. This constitutes the key recovery problem for the McEliece cryptosystem:

Key recovery problem for (Classic) McEliece. Given parameters m, r, n and a matrix $\boldsymbol{H}$ generated as in Algorithm 1, find $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ and $g \in \mathbb{F}_{2^m}[X]$ with $\deg(g) = r$ such that $\text{Gop}(\boldsymbol{x}, g)$ has parity check matrix $\boldsymbol{H}$.

In section 1, we gave an overview of some known results for key recovery for (Classic) McEliece and related cryptosystems, in particular McEliece variants where the binary Goppa codes are replaced by some other code family.

In the following, we want to explain why computing *either* a support $\boldsymbol{x}$ *or* a Goppa polynomial g is already sufficient for key recovery. In fact, being able to recover the Goppa polynomial from the support vector will be important for our key recovery attack in section 5.

Algorithm 2 Recovery of Goppa polynomial from a support vector

Input: Parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{mr \times n}$ of a proper irreducible binary Goppa code C, and a valid support vector $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ for C as an alternant code

Output: $\boldsymbol{x}' \in \mathbb{F}_{2^m}^n$ and $g \in \mathbb{F}_{2^m}[X]$ such that $\deg(g) = r$ and $C = \mathrm{Gop}(\boldsymbol{x}', g)$

Compute a basis $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n-mr}$ of C.
$L \leftarrow \{z \in \mathbb{F}_{2^m} \mid z \neq x_i \forall 1 \leq i \leq n\}$
$\mathcal{F} \leftarrow \{id\colon \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}, z \mapsto z\} \cup \{f_\alpha\colon \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}, z \mapsto \frac{1}{z-\alpha} \mid \alpha \in L\}$
for $f \in \mathcal{F}$ **do**
    $\boldsymbol{x}' = (x'_1, \ldots, x'_n) \leftarrow f(\boldsymbol{x})$
    For each $1 \leq j \leq n - mr$, compute $p_{\boldsymbol{c}_j} \coloneqq \sum_{i=1}^n c_{j,i} \prod_{j \neq i}(X - x'_i)$.
    Compute the highest common factor $h(X)$ of $p_{\boldsymbol{c}_1}, \ldots, p_{\boldsymbol{c}_{n-mr}}$.
    if $\deg(h) = 0$ **then**
        continue
    else if $h(X)$ is a square **then**
        $g(X) \leftarrow$square root of $h(X)$
        return $(\boldsymbol{x}', g(X))$
    else ▷ This should happen very rarely.
        return $(\boldsymbol{x}', h(X))$
    end if
end for

Recovering a support vector from a Goppa polynomial. This is essentially solved by the support-splitting algorithm [21]: Given the Goppa polynomial and the set of entries in the support, it recovers the support $\boldsymbol{x}$. Note that the support-splitting algorithm requires one to know the *set* of supports $\{x_i\}$. If $n = 2^m$, we know that this set is $\mathbb{F}_{2^m}$, otherwise we have to iterate through all subsets of $\mathbb{F}_{2^m}$ of size n until the correct one is found and the support-splitting algorithm succeeds. Thus, if the length n of the code is much smaller than 2^m, this significantly increases the complexity of this algorithm.

Recovering a Goppa polynomial from a support vector. Conversely, suppose that a support vector $\boldsymbol{x}$ for the Goppa code $C = \mathrm{Gop}(\boldsymbol{x}, g)$ is known, where $g \in \mathbb{F}_{2^m}[X]$ is irreducible and unknown. By Proposition 1, for any $\boldsymbol{c} \in C$, we have that

$$g(X) \text{ divides } p_{\boldsymbol{c}} \coloneqq \sum_{i=1}^{n} c_i \prod_{j \neq i}(X - x_j)$$

Hence, by computing the highest common factor of the polynomials $p_{\boldsymbol{c}}$ where we let $\boldsymbol{c}$ iterate through a basis of the given code C, we obtain a polynomial $h(X)$ of which $g(X)$ must be a factor. Then we must have $\mathrm{Gop}(\boldsymbol{x}, g) = \mathrm{Gop}(\boldsymbol{x}, h)$: The inclusion $\subset$ follows from the fact that h is constructed as the highest common factor of the polynomials $p_{\boldsymbol{c}}$, and the reverse inclusion follows from the fact that $g(X) \mid h(X)$.

In fact, we can say more about h. Since g is irreducible, Corollary 1 implies that $\mathrm{Gop}(\boldsymbol{x}, g) = \mathrm{Gop}(\boldsymbol{x}, g^2)$. Hence g^2 must divide h. In all our experiments, it

has been the case that $h = g^2$. By taking the square root in this case, we obtain g.

More generally, let us assume that $\boldsymbol{x}$ is a valid support for C as an alternant code, meaning that there is some multiplier $\boldsymbol{y}$ such that $C = \mathrm{Alt}_r(\boldsymbol{x}, \boldsymbol{y})$, but we do not assume that $\boldsymbol{x}$ is a valid support of C as a Goppa code. In our key recovery algorithm in section 5, we will need to be able to compute a valid Goppa polynomial given such $\boldsymbol{x}$. One way to do this is to iterate through all transformations described in Proposition 4, hoping that a valid support of C as a Goppa code can be obtained in this way from $\boldsymbol{x}$. For each support $\boldsymbol{x}'$ obtained in this way, we apply the approach in the previous paragraph to try to compute a corresponding Goppa polynomial. We can make this more efficient by using the observation in Proposition 5 that the Frobenius endomorphism as well as the affine transformations f map Goppa supports to Goppa supports. For this reason, it is sufficient to iterate through the identity map as well as all Möbius transformations of the form $f_\alpha(x) = \frac{1}{x-\alpha}$, where $\alpha \in \mathbb{F}_{2^m}$ with $\alpha \neq x_i$ for all $1 \leq i \leq n$.

The full algorithm is described in Algorithm 2.

3 A closer analysis of the higher-order vanishing distinguisher

In subsection 3.1, we recall the basic definition of central objects introduced for the higher-order vanishing distinguisher [13]. In subsection 3.2, we provide a more careful analysis of the structure of higher-order vanishing polynomials that we obtain in the case of binary Goppa codes. The main result is Theorem 1, which will be central to our key recovery attack. This also yields an improved version of the shortening result in [13, Proposition 9]. Many of the proofs in subsection 3.2 are very similar to proofs in [13, section 5], but we choose to reproduce them here in slightly adapted form to make the section more self-contained.

3.1 Higher-order vanishing: Definition and basic properties

Throughout this section, q is a power of 2. For any k, we denote by $R_k \coloneqq \mathbb{F}_q[X_1, \ldots, X_k]$ the polynomial ring in k indeterminates and by $R_k^{(p)}$ the $\mathbb{F}_q$-vector space of homogeneous polynomials of degree p in R_k. Furthermore, we denote by $U_k^{(p)} \subset R_k^{(p)}$ the subspace spanned by all monomials of degree p where each indeterminate occurs to degree at most 1.

The central object of the distinguisher in [13] is the following map whose kernel consists of polynomials that simultaneously vanish of higher order when evaluated at the columns of a given matrix $\boldsymbol{A}$.

Definition 4. *Let $\boldsymbol{A} \in \mathbb{F}_q^{k \times n}$ and denote by $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{F}_q^k$ the columns of $\boldsymbol{A}$. We define the $\mathbb{F}_q$-linear map*

$$\varphi_{\boldsymbol{A}}^{(p)} \colon R_k^{(p)} \to \prod_{j=1}^{n} \mathbb{F}_q^{\binom{k+p-3}{p-2}}, \quad f \mapsto \left[\frac{\partial^\ell f}{\partial X_{i_1} \cdots \partial X_{i_\ell}}(\boldsymbol{a}_j) \right]_{1 \leq i_1 \leq \ldots \leq i_\ell \leq k}^{j=1,\ldots,n;\ 0 \leq \ell \leq p-2}$$

In other words, $\varphi_{\boldsymbol{A}}^{(p)}$ computes all partial derivatives of order at most $p-2$ of a given homogeneous polynomial $f \in R_k^{(p)}$ and evaluates them at all columns of $\boldsymbol{A}$. Note that for $p=2$, the map $\varphi_{\boldsymbol{A}}^{(2)}$ simply evaluates homogeneous quadratic polynomials at all columns of $\boldsymbol{A}$. In general, the kernel of $\varphi_{\boldsymbol{A}}^{(p)}$ contains precisely those polynomials of degree p that simultaneously vanish of order $p-1$ at all points $\boldsymbol{a}_i$.

Given the parity check matrix $\boldsymbol{A}$ of either a binary Goppa code or a general linear code, the main idea of the distinguisher is to compute the dimension of the kernel of $\varphi_{\boldsymbol{A}}^{(p)}$. [13, Proposition 3] shows that this dimension only depends on the row space of $\boldsymbol{A}$. For appropriately chosen p, this kernel is trivial with high probability in the random case but non-trivial in the Goppa case.

Actually, [13] makes some modifications to $\varphi_{\boldsymbol{A}}^{(p)}$ in order to decrease the size of the range and domain and to avoid kernel elements that occur only due to the characteristic of the ground field:

- After applying row operations to $\boldsymbol{A}$ to transform it into a matrix $\boldsymbol{A}'$ in reduced row-echelon form, [13, Lemma 2] shows that the interesting part (i.e. the part that does not exist only because of the characteristic of the ground field) of the kernel of $\varphi_{\boldsymbol{A}}^{(p)}$ is contained in $U_k^{(p)}$. Therefore, we may restrict to this space and then omit the columns of the code matrix which are equal to the standard basis vectors.
- [13, Remark 4] shows that restricting to computing the partial derivatives of order $p-2^\ell$ for $1 \leq \ell \leq \lfloor \log_2 p \rfloor$ does not change the kernel of $\varphi_{\boldsymbol{A}}^{(p)}$.

This motivates the following definition (cf. [13, Definition 6]).

Definition 5. *Let $\boldsymbol{A} \in \mathbb{F}_q^{k\times n}$ be a matrix of rank k. Let $\tilde{\boldsymbol{A}} \in \mathbb{F}_q^{k\times(n-k)}$ be the matrix obtained from $\boldsymbol{A}$ by first computing the (unique) reduced row-echelon form of $\boldsymbol{A}$ and then removing the k columns that contain the pivot entries, i.e. that contain the leading 1 in a row. Let $\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_{n-k}$ denote the columns of $\tilde{\boldsymbol{A}}$. We define*

$$\tilde{\varphi}_{\boldsymbol{A}}^{(p)} \colon U_k^{(p)} \to \prod_{j=1}^{n-k} \mathbb{F}_q^{M_{k,p}}, \quad f \mapsto \left[\frac{\partial^{p-2^\ell} f}{\partial X_{i_1} \cdots \partial X_{i_\ell}}(\tilde{\boldsymbol{a}}_j) \right]_{1\leq i_1<\ldots<i_\ell\leq k}^{j=1,\ldots,n-k;\ 1\leq\ell\leq\lfloor\log_2(p)\rfloor}$$

where $M_{k,p} \coloneqq \sum_{\ell=1}^{\lfloor \log_2(p)\rfloor} \binom{k}{p-2^\ell}$.

Note that in the definition of $\tilde{\varphi}_{\boldsymbol{A}}^{(p)}$, we only compute partial derivatives of orders $p-2^\ell$ and only those taking the derivative by any fixed indeterminate at most once.

In [13], we analyse the dimension of the kernel of $\varphi_{\boldsymbol{A}}^{(p)}$ for randomly selected matrices $\boldsymbol{A}$. This analysis is crucial in order to determine, for any given parameter set, the value of p required in order to distinguish Goppa codes from random general linear codes. The result is the following heuristic, which is supported by extensive experimental evidence (see [13, Heuristic 1]).

Heuristic 2. Suppose $\boldsymbol{A} \in \mathbb{F}_q^{k \times n}$ is randomly selected. Let $\tilde{n} \coloneqq n - k > k$. If k is sufficiently large, then with high probability,

$$\dim_{\mathbb{F}_q}\left(\ker\left(\tilde{\varphi}_{\boldsymbol{A}}^{(p)}\right)\right) = \text{degree } p \text{ coefficient of } \left[(1+t)^k\left(1-t^2\right)^{\tilde{n}}\right]^+$$

where, for a polynomial $\zeta(t) = \sum_{i=0}^{N} a_i t^i \in \mathbb{Z}[t]$ where $d \in \mathbb{N} \cup \{\infty\}$ is the smallest index such that $a_d < 0$, we write $[\zeta]^+ \coloneqq \sum_{i=0}^{N} a_i' \cdot t^i$ with

$$a_i' = \begin{cases} a_i \text{ if } i < d \\ 0 \text{ otherwise} \end{cases}$$

For given parameters, this heuristic allows to compute the smallest degree p such that the kernel of $\tilde{\varphi}_{\boldsymbol{A}}^{(p)}$ is trivial with high probability for randomly selected matrices. For our key recovery, for a given code parameter set we will choose such a p and then exploit the structure of non-trivial elements in the kernel of $\tilde{\varphi}_{\boldsymbol{H}}^{(p)}$ for a parity check matrix $\boldsymbol{H}$ of the given Goppa code. In the following section, we analyse the structure of these elements more closely.

3.2 Structure of higher-order vanishing polynomials for shortened dual Goppa codes

Suppose that $C = \mathrm{Gop}(\boldsymbol{x}, g)$ is a binary proper irreducible Goppa code of degree r and field extension degree m. By Proposition 2, we have

$$\mathrm{span}_{\mathbb{F}_{2^m}}\left(C^\perp\right) = \sum_{i=0}^{m-1} \mathrm{GRS}_r\left(\boldsymbol{x}, \boldsymbol{y}\right)^{(2^i)}$$

where $\boldsymbol{y} \coloneqq \frac{1}{g(\boldsymbol{x})}$. A generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ is given by

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}} \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2)} \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^2)} \\ \cdots \\ (\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^{m-1})} \end{pmatrix} \text{ with } \boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_1 x_1 & y_2 x_2 & \cdots & y_n x_n \\ y_1 x_1^2 & y_2 x_2^2 & \cdots & y_n x_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ y_1 x_1^{r-1} & y_2 x_2^{r-1} & \cdots & y_n x_n^{r-1} \end{pmatrix} \tag{3}$$

We want to analyse the kernel of $\tilde{\varphi}_{\boldsymbol{A}}^{(p)}$. It is convenient to label the indeterminates of the polynomial ring in the domain of $\tilde{\varphi}_{\boldsymbol{A}}^{(p)}$ by the indeterminates in the tuple

$$\boldsymbol{X} = (X_0^{(0)}, X_1^{(0)}, \ldots, X_{r-1}^{(0)}, X_0^{(1)}, \ldots, X_{r-1}^{(m-1)}) \tag{4}$$

where the indeterminate $X_i^{(u)}$ corresponds to row i in the block $(\boldsymbol{G}_{\boldsymbol{x},\boldsymbol{y}})^{(2^u)}$ of the generator matrix $\boldsymbol{A}$.

Later we will consider a suitable shortening of this code. [13, section 5.3] already contains such an analysis. However, we will improve on that result by

showing that we can shorten in more positions while still obtaining a non-trivial kernel. This tighter result will be important for our key recovery attack.

The key to this analysis and obtaining a tighter result is a more careful choice of basis for the code $\mathrm{span}_{\mathbb{F}_{2^m}}\left(C^{\perp}\right)$. To do this, we prove a preparatory lemma.

Lemma 2. *Suppose $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ has pairwise distinct entries and $g \in \mathbb{F}_{2^m}[X]$ is square-free of degree r. Let $\boldsymbol{y} \coloneqq \frac{1}{g(\boldsymbol{x})}$. Then for any $0 \leq e < m-1$, the elements*

$$\boldsymbol{a}_i^{(j)} \coloneqq (\boldsymbol{y}^2\boldsymbol{x}^{2i+1})^{2^j} \text{ for } 0 \leq i < r, 0 \leq j < e, \text{ and } \boldsymbol{b}_k \coloneqq (\boldsymbol{y}^2\boldsymbol{x}^k)^{2^e} \text{ for } 0 \leq k < 2r$$

span the vector space

$$W_0 \coloneqq \sum_{i=0}^{e+1} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{2^i}$$

Thus, if this sum is direct, then the above vectors are a basis of W_0.

Proof. From Lemma 1, we deduce that

$$W_0 = \sum_{i=0}^{e+1} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{2^i} = \sum_{i=0}^{e} \mathrm{GRS}_{2r}(\boldsymbol{x}, \boldsymbol{y}^2)^{2^i}$$

This sum is not direct of course. We see from this that $\boldsymbol{a}_i^{(j)} \in W_0$ and $\boldsymbol{b}_k \in W_0$. We need to show that they span W_0. It is sufficient to show that any element of the form $(\boldsymbol{y}^2\boldsymbol{x}^{2i})^{2^j}$ for $0 \leq i < r$ and $0 \leq j < e$ can be written as a linear combination of the $\boldsymbol{a}_i^{(j)}$ and $\boldsymbol{b}_k$. Suppose this is not the case, and let $\boldsymbol{c} = (\boldsymbol{y}^2\boldsymbol{x}^{2i})^{2^j}$ be an element with maximal j such that $\boldsymbol{c} \notin \mathrm{span}(\boldsymbol{a}_i^{(j)}, \boldsymbol{b}_k)$. We must have $j < e$ since otherwise, $\boldsymbol{c}$ is equal to one of the $\boldsymbol{b}_k$. Writing $g(X) = \sum_{\ell=0}^{r} g_\ell \cdot X^\ell$, we have

$$\boldsymbol{c} = (\boldsymbol{y}^2\boldsymbol{x}^{2i})^{2^j} = \left(\frac{g(\boldsymbol{x})^2\boldsymbol{x}^{2i}}{g(\boldsymbol{x})^4}\right)^{2^j} = \left(\frac{g(\boldsymbol{x})\boldsymbol{x}^i}{g(\boldsymbol{x})^2}\right)^{2^{j+1}} = \sum_{\ell=0}^{r} g_\ell^{2^{j+1}} \cdot (\boldsymbol{y}^2\boldsymbol{x}^{\ell+i})^{2^{j+1}}$$

Now every term $(\boldsymbol{y}^2\boldsymbol{x}^{\ell+i})^{2^{j+1}}$ where $\ell + i$ is odd must be in $\mathrm{span}_{\mathbb{F}_{2^m}}(\boldsymbol{a}_i^{(j)}, \boldsymbol{b}_k)$ because it is equal to $\boldsymbol{a}_{(\ell+i-1)/2}^{(j+1)}$ if $j+1 < e$ or to $\boldsymbol{b}_{\ell+i}$ if $j+1 = e$. If $\ell+i$ is even, then $(\boldsymbol{y}^2\boldsymbol{x}^{\ell+i})^{2^{j+1}}$ must also be in the span by the minimal choice of j. But then $\boldsymbol{c}$ is in the span, which is a contradiction. This proves the assertion. □

In the following, we let

$$e \coloneqq 1 + \lfloor \log_2((r-1)/(p-1)) \rfloor \tag{5}$$

It will later become clear why we choose e in this way. We choose a generator matrix $\boldsymbol{A}_0 \in \mathbb{F}_{2^m}^{mr \times n}$ of $\mathrm{span}_{\mathbb{F}_{2^m}}\left(C^{\perp}\right)$ as follows: For the first $r \cdot (e+2)$ rows, we set the vectors

$$\boldsymbol{a}_0^{(0)}, \ldots, \boldsymbol{a}_{r-1}^{(0)}, \boldsymbol{a}_0^{(1)}, \ldots, \boldsymbol{a}_{r-1}^{(e-1)}, \boldsymbol{b}_0, \ldots, \boldsymbol{b}_{2r-1}$$

as defined in Lemma 2. We set the remaining rows of $\boldsymbol{A}_0$ in an arbitrary way by basis completion such that the rows of $\boldsymbol{A}_0$ form a basis of $\mathrm{span}_{\mathbb{F}_{2^m}}\left(C^{\perp}\right)$.

Structure of the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$. We now want to analyse the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$. Recall that the domain of this map is the space of homogeneous polynomials of degree p in mr (which is the number of rows in $\boldsymbol{A}_0$) indeterminates. It is convenient to name the indeterminates in a way that the correspondence to the rows of $\boldsymbol{A}_0$ becomes clear. Therefore, we introduce the indeterminates

$$\begin{aligned} Z_i^{(j)} &\text{ corresponding to } \boldsymbol{a}_i^{(j)} \text{ for } 0 \le i < r, 0 \le j < e \\ Z_k &\text{ corresponding to } \boldsymbol{b}_k \text{ for } 0 \le k < 2r \\ Z'_\ell &\text{ corresponding to the remaining rows of } \boldsymbol{A}_0 \text{ for } 0 \le \ell < mr - (e+2)r \end{aligned}$$

as well as the tuple containing these indeterminates

$$\boldsymbol{Z} = (\boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{Z}_2)$$

with $\boldsymbol{Z}_0 = (Z_0^{(0)}, Z_1^{(0)}, \dots, Z_{r-1}^{(0)}, Z_0^{(1)}, \dots, Z_{r-1}^{(e-1)})$, $\boldsymbol{Z}_1 = (Z_0, \dots, Z_{2r-1})$ and $\boldsymbol{Z}_3 = (Z'_0, \dots, Z'_{mr-(e+2)r-1})$.

We define the matrices

$$\boldsymbol{B}_u^{(p)} \coloneqq \begin{pmatrix} Z_0^{(e-u)} & Z_1^{(e-u)} & \dots & Z_{r-2^{u-1}(p-1)-1}^{(e-u)} \\ Z_{2^{u-1}}^{(e-u)} & Z_{1+2^{u-1}}^{(e-u)} & \dots & Z_{r-2^{u-1}(p-2)-1}^{(e-u)} \\ \dots & \dots & \dots & \dots \\ Z_{2^{u-1}(p-1)}^{(e-u)} & Z_{1+2^{u-1}(p-1)}^{(e-u)} & \dots & Z_{r-1}^{(e-u)} \end{pmatrix}$$

and

$$\boldsymbol{B}^{(p)} \coloneqq \begin{pmatrix} Z_0 & Z_1 & \dots & Z_{2r-p} \\ Z_1 & Z_2 & \dots & Z_{2r-p+1} \\ \dots & \dots & \dots & \dots \\ Z_{p-1} & Z_p & \dots & Z_{2r-1} \end{pmatrix}$$

as well as

$$\boldsymbol{\Phi}^{(p)} \coloneqq (\boldsymbol{B}_e^{(p)} \mid \boldsymbol{B}_{e-1}^{(p)} \mid \dots \mid \boldsymbol{B}_1^{(p)} \mid \boldsymbol{B}^{(p)})$$

Note that by the choice of e in Equation 5, all the indices i in the indeterminates $Z_i^{(j)}$ that occur as entries in $\boldsymbol{B}_u^{(p)}$ for $1 \le u \le e$ actually lie in the range $0 \le i < r$, so these matrices are well-defined.

$\boldsymbol{B}_u^{(p)}$ is a $p \times (r - 2^{u-1}(p-1))$ matrix, and $\boldsymbol{B}^{(p)}$ is a $p \times (2r - p + 1)$ matrix. Hence $\boldsymbol{\Phi}^{(p)}$ is a $p \times ((e+2)r - 2^e(p-1)$ matrix.

The following observation is the key reason why we constructed the matrix $\boldsymbol{\Phi}^{(p)}$ as specified.

Lemma 3. *Suppose $x, y \in \mathbb{F}_{2^m}$. If we substitute $Z_i^{(v)} \mapsto (y^2 x^{2i+1})^{2^v}$ for all $0 \le i < r$ and $0 \le v < m$ as well as $Z_k \mapsto (y^2 x^k)^{2^e}$ for all $0 \le k < 2r$ in the matrix $\boldsymbol{\Phi}^{(p)}$, the resulting matrix has rank 1 over $\mathbb{F}_{2^m}$.*

Proof. One can easily check that if we make these substitutions in any of the $\boldsymbol{B}_u^{(p)}$ or in $\boldsymbol{B}^{(p)}$, then any row can be obtained from the previous one via multiplication by x^{2^e}. This immediately implies the claim. □

Using this, we can now describe non-trivial elements in the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$. In particular, this yields a lower bound for the dimension of this kernel.

Proposition 6. *The $p \times p$ minors of $\boldsymbol{\Phi}^{(p)}$ are linearly independent and are in the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$.*

Proof. We prove linear independence by induction on p and first introduce some notation. Let n_{col} be the number of columns of $\boldsymbol{\Phi}^{(p)}$. Given a set $S \subset \{1, \ldots, n_{col}\}$ with $|S| = p$, we denote by $\boldsymbol{\Phi}^{(p)}_S$ the $p \times p$ submatrix formed by the columns of $\boldsymbol{\Phi}^{(p)}$ with indices in S.

The entries in any fixed row of $\boldsymbol{\Phi}^{(p)}$ are distinct indeterminates of the polynomial ring and therefore linearly independent. Now suppose that $p \geq 2$ and suppose for contradiction that the $p \times p$ minors are not linearly independent. Let a non-trivial linear combination be given by

$$\sum_{i=1}^{\ell} \lambda_i \cdot \det\left(\boldsymbol{\Phi}^{(p)}_{S_i}\right) = 0 \tag{6}$$

where $\ell \geq 1$, $\lambda_i \in \mathbb{F}_{q^m} \backslash \{0\}$ for all $1 \leq i \leq \ell$ and the $S_i \subset \{1, \ldots, n_{col}\}$ are distinct subsets of size p. Let $a \in \{1, \ldots, n_{col}\}$ be the largest column index contained in some S_i, and let $S_{i_1}, \ldots, S_{i_t}$ be those sets that contain a. By maximality of a, the indeterminate (let us call it Y) in the bottom rightmost entry of the matrices $\boldsymbol{\Phi}^{(p)}_{S_{i_d}}$ for $1 \leq d \leq t$ only occurs in this entry and nowhere else in the matrices $\boldsymbol{\Phi}^{(p)}_{S_i}$ for $1 \leq i \leq \ell$. Hence using Laplace expansion of the determinants in Equation 6 by the last row and comparing coefficients of Y, we deduce that

$$\sum_{d=1}^{t} \lambda_{i_d} \cdot \det\left(\hat{\boldsymbol{\Phi}}^{(p)}_{S_{i_d} \backslash \{a\}}\right) = 0$$

where $\hat{\boldsymbol{\Phi}}^{(p)}$ denotes the matrix $\boldsymbol{\Phi}^{(p)}$ with the last row deleted. By induction, we deduce that $\lambda_{i_d} = 0$ for $1 \leq d \leq t$, contradicting the assumption that $\lambda_i \neq 0$ for all i. This concludes our proof of linear independence.

Note that by the rules for differentiating a determinant, any partial derivative of order $0 \leq \ell \leq p-2$ of a maximal minor of $\boldsymbol{\Phi}^{(p)}$ is a linear combination of $p - \ell$ minors of $\boldsymbol{\Phi}^{(p)}$ of size at least 2×2. But by Lemma 3, any such minor vanishes when evaluated at any column of $\boldsymbol{A}_0$. This implies that any partial derivative of order $0 \leq \ell \leq p-2$ of any given maximal minor of $\boldsymbol{\Phi}^{(p)}$ vanishes when evaluated at any column of $\boldsymbol{A}_0$. Thus, any maximal minor of $\boldsymbol{\Phi}^{(p)}$ is in the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$. □

Structure of the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}}$. In order to deduce from this a lower bound on the dimension of the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$, we need to consider the behaviour of $\boldsymbol{\Phi}^{(p)}$ under row operations on the matrix $\boldsymbol{A}_0$, which corresponds to a linear change of variables in the polynomials in the domain of $\varphi^{(p)}_{\boldsymbol{A}_0}$. To this end, let $\boldsymbol{S} \in \mathbb{F}_{q^m}^{mr \times mr}$

be an invertible matrix such that $\tilde{\boldsymbol{A}} \coloneqq \boldsymbol{S}\boldsymbol{A}_0$ is in reduced row echelon form. We define the change of variables ring homomorphism

$$\alpha_{\boldsymbol{S}^{-1}} \colon \mathbb{F}_{2^m}[\boldsymbol{Z}] \to \mathbb{F}_{2^m}[\boldsymbol{Y}], \quad f(\boldsymbol{Z}) \mapsto f(\boldsymbol{Y} \cdot (\boldsymbol{S}^{-1})^T)$$

where $\boldsymbol{Z} = (\boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{Z}_2)$ as defined above and $\boldsymbol{Y} = (Y_1, \ldots, Y_{mr})$ are taken to be row vectors each with mr indeterminates as entries. We denote by $\tilde{\boldsymbol{\Phi}}^{(p)}$ the matrix obtained from $\boldsymbol{\Phi}^{(p)}$ by applying $\alpha_{\boldsymbol{S}^{-1}}$ to each entry. In the following, we let $\mathbb{F}_{2^m}[\boldsymbol{Y}]$ be the domain of $\varphi^{(p)}_{\boldsymbol{S}\boldsymbol{A}_0}$ to distinguish it from the domain of $\varphi^{(p)}_{\boldsymbol{A}_0}$ (which we took to be $\mathbb{F}_{2^m}[\boldsymbol{Z}]$). This is just a naming convention for the indeterminates.

We can now identify non-trivial elements in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$:

Proposition 7. *The $p \times p$ minors of $\tilde{\boldsymbol{\Phi}}^{(p)}$ are in the kernel of $\varphi^{(p)}_{\tilde{\boldsymbol{A}}}$ and are linearly independent. Furthermore, in any $p \times p$ minor, each indeterminate Y_j occurs to degree at most 1.*

Proof. The fact that the $p \times p$ minors of $\tilde{\boldsymbol{\Phi}}^{(p)}$ are linearly independent follows immediately from the fact that the $p \times p$ minors of $\boldsymbol{\Phi}^{(p)}$ are linearly independent (see Proposition 6) and that the former and latter differ only by a linear change of variables. The chain rule for differentiation and the fact that the $p \times p$ minors of $\boldsymbol{\Phi}^{(p)}$ are in the kernel of $\varphi^{(p)}_{\boldsymbol{A}_0}$ imply that the $p \times p$ minors of $\tilde{\boldsymbol{\Phi}}^{(p)}$ are in the kernel of $\varphi^{(p)}_{\tilde{\boldsymbol{A}}}$.

Now fix a $p \times p$ submatrix $\boldsymbol{M}$ of $\tilde{\boldsymbol{\Phi}}^{(p)}$. We need to show that in $\det(\boldsymbol{M})$, each indeterminate occurs to degree at most 1. The entries of $\boldsymbol{M}$ are homogeneous linear polynomials in the indeterminates. Consider a fixed but arbitrary indeterminate Y_j. The standard basis vector $\boldsymbol{e}_j$ occurs as a column in $\boldsymbol{S}\boldsymbol{A}_0$, so $\tilde{\boldsymbol{\Phi}}^{(p)}$ and therefore $\boldsymbol{M}$ has rank at most 1 when evaluated at $\boldsymbol{e}_j$ by Lemma 3. Hence the matrix formed by the coefficients of Y_j in $\boldsymbol{M}$ has rank at most 1. This means that using elementary column operations on $\boldsymbol{M}$, one can transform it such that Y_j occurs only in a single column. The determinant of such a matrix is clearly linear in Y_j. Since elementary column operations do not alter the determinant, this implies that the same holds for $\boldsymbol{M}$. □

Note that by definition of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$, the above proposition shows that the $p \times p$ minors of $\tilde{\boldsymbol{\Phi}}^{(p)}$ are contained in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$.

Structure of the kernel after shortening. We have identified many linearly independent elements in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$. In this section, we want to obtain a similar result for the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s]}}$, where $\boldsymbol{A}_0^{[s]}$ is the generator matrix of the code represented by the generator matrix $\boldsymbol{A}$ (and equivalently by $\boldsymbol{A}_0$), but shortened in a certain number of positions. We want to determine in how many positions we can shorten while still obtaining non-trivial elements in the kernel.

By reordering the columns of $\boldsymbol{A}_0$, we may assume that we shorten the code with generator matrix $\boldsymbol{A}_0$ in the first s positions. Let $\boldsymbol{S} \in \mathbb{F}_{2^m}^{mr \times mr}$ be an invertible matrix such that $\boldsymbol{H} = \boldsymbol{S}\boldsymbol{A}_0$ is in reduced row-echelon form. Notice we may obtain a generator matrix $\boldsymbol{A}_0^{[s]}$ of the shortened code by removing the first $s' \leq s$ rows of $\boldsymbol{H}$ that contain a leading 1 in the first s columns. Thus, the map $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s]}}$ can be regarded as the restriction of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0}$ to the subalgebra generated by indeterminates $Y_{s'+1}, \ldots, Y_{mr}$.

Remark 2. In our case of the code $\mathrm{span}_{\mathbb{F}_{2^m}}(C^{\perp})$ where C is a proper binary Goppa code, if we randomly choose s positions where we shorten and s is significantly smaller than the dimension of $C^{\perp}$, it is likely that the corresponding columns of the generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^{\perp})$ are linearly independent and that thus $s = s'$.

In the following, we show that we can shorten in

$$s_* \coloneqq (e+2) \cdot r - (p-1) \cdot 2^e - p \tag{7}$$

positions, where $e \coloneqq 1 + \lfloor \log_2((r-1)/(p-1)) \rfloor$ is as before.

Proposition 8. *The kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s_*]}}$ contains a non-zero polynomial $h(\boldsymbol{Y}) \in \mathbb{F}_{2^m}[\boldsymbol{Y}]$. It has the property that $h(\boldsymbol{Z} \cdot \boldsymbol{S}^T) \in \mathbb{F}_{2^m}[\boldsymbol{Z}]$ depends only on indeterminates $Z_i^{(u)}$ for $0 \leq u < e$ and $0 \leq i < r$ and on Z_j for $0 \leq j < 2r$.*

Proof. Suppose without loss of generality that we shorten in the first s_* positions. We denote by $n_{\mathrm{col}} \coloneqq (e+2) \cdot r - (p-1) \cdot 2^e$ the number of columns of $\tilde{\boldsymbol{\Phi}}^{(p)}$.

Since $\boldsymbol{H}$ contains the standard unit vectors as columns and $\tilde{\boldsymbol{\Phi}}^{(p)}$ has rank 1 when evaluated at the columns of $\boldsymbol{H}$, the following holds: For any indeterminate Y_i, the matrix formed by the coefficients of Y_i in $\tilde{\boldsymbol{\Phi}}^{(p)}$ has rank 1. In particular, this holds for $Y_1, \ldots, Y_{s_*}$. Therefore, using elementary column operations, we may transform $\tilde{\boldsymbol{\Phi}}^{(p)}$ to a matrix $\tilde{\boldsymbol{\Phi}}'^{(p)}$ such that its first $n_{\mathrm{col}} - s_* = p$ columns only contain the indeterminates $Y_{s_*+1}, \ldots, Y_{mr}$. Thus, the $p \times p$ minor obtained from the first p columns of $\tilde{\boldsymbol{\Phi}}'^{(p)}$ contains only those indeterminates. Set $h(\boldsymbol{Y})$ to be this $p \times p$ minor. Since elementary column operations of $\tilde{\boldsymbol{\Phi}}^{(p)}$ leave the space of maximal minors invariant, Proposition 7 implies that $h(\boldsymbol{Y})$ is a non-trivial element in the kernel of $\tilde{\varphi}_{\boldsymbol{A}_0}$. It is also a non-trivial element in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s_*]}}$ since it only involves the indeterminates $Y_{s_*+1}, \ldots, Y_{mr}$.

The preimage of $h(\boldsymbol{Y})$ under the change of variables $\alpha_{\boldsymbol{S}^{-1}}$ is given by $h(\boldsymbol{Z} \cdot \boldsymbol{S}^T) \in \mathbb{F}_{2^m}[\boldsymbol{Z}]$. Since $h(\boldsymbol{Y})$ is a linear combination of $p \times p$ minors of $\tilde{\boldsymbol{\Phi}}^{(p)}$, the polynomial $h(\boldsymbol{Z} \cdot \boldsymbol{S}^T)$ is a linear combination of $p \times p$ minors of $\boldsymbol{\Phi}^{(p)}$. Hence it only depends on the indeterminates $Z_i^{(u)}$ for $0 \leq u < e$ and $0 \leq i < r$ and on Z_j for $0 \leq j < 2r$ as claimed. □

At the beginning of this section, we started with a convenient choice of basis for

$$W_0 = \sum_{i=0}^{e+1} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

and used it to, in the end, construct a non-trivial element in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s_*]}}$ with properties as described in Proposition 8. In the same way, we can start out with any of the m 'shifted' subspaces

$$W_\ell = \sum_{i=\ell}^{\ell+e+1} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

and construct an element in the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{A}_0^{[s_*]}}$ as in Proposition 8 for each of them, obtaining m elements in total. It is more convenient to phrase this in terms of the basis of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ formed by the rows of the generator matrix $\boldsymbol{A}$ introduced at the beginning of this section.

Theorem 1. *Let $C = \mathrm{Gop}(\boldsymbol{x}, g)$ be a proper irreducible binary Goppa code. Let $\boldsymbol{H} \in \mathbb{F}_2^{mr\times n}$ be the generator matrix of $C^\perp$ in reduced row-echelon form, let $\boldsymbol{A}$ be the generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ defined in Equation 3 and let $\boldsymbol{R} \in \mathbb{F}_{2^m}^{rm\times rm}$ be the invertible matrix such that $\boldsymbol{H} = \boldsymbol{R}\boldsymbol{A}$. Let*

$$s_* \coloneqq (e+2)\cdot r - (p-1)\cdot 2^e - p \text{ where } e \coloneqq 1 + \left\lfloor \log_2\left(\frac{r-1}{p-1}\right)\right\rfloor \tag{8}$$

and $\boldsymbol{H}^{[s_*]}$ be the matrix obtained from $\boldsymbol{H}$ by omitting the rows which have a leading 1 in the first s_* columns. (Note that $\boldsymbol{H}^{[s_*]}$ is a generator matrix of the code $C^\perp$ shortened in the first s_* positions.) Then over $\mathbb{F}_{2^m}$, the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ contains polynomials

$$h_0(\boldsymbol{Y}), \ldots, h_{m-1}(\boldsymbol{Y}) \in \mathbb{F}_{2^m}[Y_{s_*+1}, \ldots, Y_{mr}]^{(p)} \subset \mathbb{F}_{2^m}[\boldsymbol{Y}]^{(p)}$$

such that the following properties hold:

- *$h_\ell \neq 0$ for all $0 \leq \ell < m$*
- *h_ℓ is the determinant of a $p \times p$ matrix whose entries are linear forms in the indeterminates $Y_{s_*+1}, \ldots, Y_{mr}$.*
- *For every ℓ, the polynomial $h_\ell(\boldsymbol{X}\cdot\boldsymbol{R}^T)$ only depends on the indeterminates $X_i^{(u)}$ with $\ell \leq u \leq \ell + e + 1$ and $0 \leq i < r$, where $\boldsymbol{X}$ is the tuple of indeterminates defined in Equation 4.*

Proof. Let $\boldsymbol{T} \in \mathbb{F}_{2^m}^{mr\times mr}$ be the invertible matrix such that $\boldsymbol{A}_0 = \boldsymbol{T}\boldsymbol{A}$. By definition of $\boldsymbol{A}$ and $\boldsymbol{A}_0$, the matrix $\boldsymbol{T}$ is block-diagonal with one block of size $r(e+2)\times r(e+2)$ and one block of size $r(m-e-2)\times r(m-e-2)$. Furthermore, $\boldsymbol{R} = \boldsymbol{S}\boldsymbol{T}$.

Set $h_0(\boldsymbol{Y})$ to be the polynomial h defined in Proposition 8. Then we know that $h(\boldsymbol{Z}\cdot\boldsymbol{S}^T) \in \mathbb{F}_{2^m}[\boldsymbol{Z}]$ depends only on indeterminates $Z_i^{(u)}$ for $0 \leq u < e$ and $0 \leq i < r$ and on Z_j for $0 \leq j < 2r$. By the block-diagonal shape of $\boldsymbol{T}$, the polynomial $h(\boldsymbol{X}\cdot\boldsymbol{T}^T\boldsymbol{S}^T) = h(\boldsymbol{X}\cdot\boldsymbol{R}^T)$ then only depends on indeterminates $X_i^{(u)}$ with $0 \leq u \leq e+1$. This proves the claim for h_0.

As indicated above, the polynomials $h_1, \ldots, h_{m-1}$ can be constructed in a similar way by starting out with the shifted subspaces W_ℓ instead of W_0, and the claims follow analogously. □

Table 2. Required degree p and number of shortened positions s_* (according to Theorem 1) for distinguishing Goppa codes with indicated Classic McEliece parameters from random linear codes using the higher-order vanishing distinguisher. N and M denote the number of rows and columns, respectively, of the matrix representing $\tilde{\varphi}^{(p)}$. κ_W is the term inside $\mathcal{O}(\cdot)$ in [13, Proposition 10].

(n, r, m)	$(3488, 64, 12)$	$(4608, 96, 13)$	$(6688, 128, 13)$	$(6960, 119, 13)$	$(8192, 128, 13)$
p	22	46	52	41	39
s_*	150	158	256	275	321
$\log_2(N)$	140.1	278.1	324.5	264.9	258.2
$\log_2(M)$	133.5	270.9	316.9	257.4	250.6
κ_W	2^{297}	2^{594}	2^{690}	2^{564}	2^{550}

This result shows that one can usually shorten the duals of binary Goppa codes slightly more than predicted by [13, Proposition 9] while still obtaining a non-trivial kernel of $\tilde{\varphi}^{(p)}$. If the kernel of $\tilde{\varphi}^{(p)}$ for random linear codes of the same length and dimension is trivial with high probability, then we can distinguish such Goppa codes from random linear codes. We therefore make the following definition.

Definition 6. *A proper irreducible binary Goppa code of length n, field extension degree m and degree r is called* p-distinguishable *if the degree p coefficient of*

$$\left[(1+t)^{mr-s_*}\left(1-t^2\right)^{n-mr}\right]^+$$

is zero, where s_* is as defined in Equation 8.

Using Theorem 1, we obtain slightly improved estimates of the complexity of the higher-order vanishing distinguisher applied to Classic McEliece parameter sets, see Table 2.

In fact, in all our experiments, it turned out that the polynomials $h_0, \ldots, h_{m-1}$ identified in Theorem 1 span the entire kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ if $p \geq 3$ and the given Goppa code was p-distinguishable[2]. So for $p \geq 3$, the result for the shortening that we described above is tight in the sense that it appears to be exactly the number of positions we can shorten just before the kernel of $\tilde{\varphi}^{(p)}$ becomes trivial. We summarise our observation in the following heuristic observation. We shall see that this observation is central to our key recovery algorithm.

Heuristic Assumption 1. Let $p > 2$ and $\boldsymbol{H} \in \mathbb{F}_2^{rm \times n}$ be the parity check matrix of a p-distinguishable proper irreducible binary Goppa code in reduced row-echelon form. Let $\boldsymbol{H}^{[s_*]}$ be the matrix obtained from $\boldsymbol{H}$ by deleting the first s_* (defined in Equation 8) rows. Then the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ has dimension m.

[2] If the code is not p-distinguishable, then of course we expect many more elements in the kernel both for Goppa and for random linear codes with the same parameters.

Table 3. Dimension of the kernel of $\tilde{\varphi}^{(2)}_{\boldsymbol{A}_s}$ for different numbers s of shortened positions, where $\boldsymbol{A}_s$ is the generator matrix of the shortened dual of a binary Goppa code with (in the left table) $m = 9$, $r = 3$ and $n = 512$, and (in the right table) $m = 10$, $r = 6$ and $n = 1024$. The shortening predicted by Theorem 1 is in bold.

s	0	1	2	3	4	5	**6**	7
$\dim\left(\ker\left(\tilde{\varphi}^{(2)}\right)\right)$	162	136	111	87	64	42	21	2

s	18	19	**20**	21	22	23
$\dim\left(\ker\left(\tilde{\varphi}^{(2)}\right)\right)$	201	160	120	81	43	6

Remark 3. We have verified this heuristic observation by extensive computation for many different parameter sets of binary Goppa codes for $p = 3$ and $p = 4$. However, it fails for $p = 2$. Note that in degree $p = 2$, our distinguisher reduces to the well-known square distinguisher. See Table 3 for examples of how the dimension of the kernel of $\tilde{\varphi}^{(2)}$ behaves under shortening. It seems that the structure of the kernel of $\tilde{\varphi}^{(2)}$ is more complicated than for $p \geq 3$. [17, section 4] identifies five types of elements in the kernel of $\varphi^{(2)}_{\boldsymbol{H}}$ where $\boldsymbol{H}$ is a parity check matrix of a binary Goppa code. Analogous elements of what are called Types 1-4 also appear in the kernel of $\varphi^{(p)}_{\boldsymbol{H}}$ for higher p and are taken into account by our construction of elements above. However, elements of Type 5 seem to be quite different from the other types, and one possible explanation is that such elements do not exist in the kernel of $\varphi^{(p)}_{\boldsymbol{H}}$ for $p \geq 3$. It would be interesting to investigate this in more detail.

4 Essential variables of homogeneous polynomials

In the previous section, we saw that the kernel of $\tilde{\varphi}^{(p)}$ applied to a suitable shortening of the dual of a binary Goppa code contains polynomials which, up to a linear change of variables, depend only on few variables. For our key recovery attack, it will be central to compute such polynomials. Therefore, we need a criterion to recognise when, after a suitable change of variables, a polynomial depends only on few variables. In subsection 4.1, we present such a general criterion, and in subsection 4.2, we apply it to our specific situation.

4.1 General definitions and result

Let F be a field and $S = F[X_1, \ldots, X_n]$ be the polynomial ring over F in n indeterminates. [6] considers the following question: Given a homogeneous polynomial $f \in S$, what is the minimal number of linear forms required such that f can be expressed in terms of them? Let us recall the following definition from [6].

Definition 7. *Let $f \in S$ be a homogeneous polynomial. The* essential number of variables of f*, denoted by $N_{\mathrm{ess}}(f)$, is the smallest integer r for which there exist linear forms (i.e. homogeneous polynomials of degree one) $y_1, \ldots, y_r \in S$ such that $f \in F[y_1, \ldots, y_r]$.*

We need a criterion to efficiently compute or at least bound the number of essential variables of a given polynomial. It is shown in [6] that in characteristic 0, the essential number of variables of a polynomial can be easily computed using the following matrix.

Definition 8. *Let $f \in S$ be homogeneous of degree d and fix a monomial basis $M_1, \ldots, M_N$ of the vector space of homogeneous polynomials in S of degree $d-1$. For each $1 \leq i \leq n$, we write*

$$\frac{\partial f}{\partial X_i} = \sum_{j=1}^{n} c_{i,j} M_j$$

where $c_{i,j} \in F$. The first catalecticant matrix $\mathcal{C}_f$ of f *is defined as the $n \times N$ matrix $\mathcal{C}_f$ over F with*

$$(\mathcal{C}_f)_{i,j} = c_{i,j}$$

Proposition 9 ([6], Proposition 1). *Suppose* $\mathrm{char}(F) = 0$ *and $f \in S$ is homogeneous. Then $N_{\mathrm{ess}}(f) = \mathrm{rk}(\mathcal{C}_f)$.*

Note that the above Proposition is clearly not true in positive characteristic. For example, in characteristic 2, all partial derivatives of $f = X_1^2$ vanish, but the essential number of variables is 1 and not 0. Even if we restrict to polynomials where each indeterminate occurs to degree at most 1, the result is not true: For $f = X_1X_2 + X_1X_3 + X_2X_3$ and characteristic 2, we have $\mathrm{rk}(\mathcal{C}_f) = 2$, but one can check that $N_{\mathrm{ess}}(f) = 3$. However, at least we still have the following inequality:

Proposition 10. *Let F be a field of arbitrary characteristic and $f \in S$ be homogeneous. Then* $\mathrm{rk}(\mathcal{C}_f) \leq N_{\mathrm{ess}}(f)$.

Proof. Before going into the proof, we introduce a linear pairing that will be useful. Let S_1 be the F-vector space of linear forms in S. We denote by $\partial_1, \ldots, \partial_n$ the operators acting on polynomials in S by first-order partial differentiation by variables $X_1, \ldots, X_n$ respectively, and we let T_1 be the n-dimensional F-vector space of formal F-linear combinations of those operators. Thus, the action of T_1 on elements of S_1 via partial differentiation defines a bilinear pairing

$$T_1 \times S_1 \to F, \quad (a_1\partial_1 + \ldots + a_n\partial_n, g) \mapsto a_1 \frac{\partial g}{\partial X_1} + \ldots + a_n \frac{\partial g}{\partial X_n}$$

Note that this pairing is well-defined since first-order partial derivatives of linear forms are elements in the field. The pairing is the usual inner product in F^n of the vector formed by the coefficients a_i of the differential operators with the vector formed by the coefficients of the variables X_i in g. Therefore, it is clearly non-degenerate.

Now let r be the essential number of variables of f. Then there exist linear forms $y_1, \ldots, y_r \in S_1$ such that $f = F[y_1, \ldots, y_r]$. Hence there are at least $n - r$ linearly independent linear forms $L_1, \ldots, L_{n-r} \in T_1$ that are orthogonal to all the y_i with respect to the pairing above, i.e. $L_i(y_j) = 0$ for all $1 \leq i \leq n - r$ and

$1 \leq j \leq r$. Writing $L_i = l_{i,1}\partial_1 + \ldots + l_{i,n}\partial_n$, by the chain rule for differentiation, we deduce that

$$L_i(f) = l_{i,1}\frac{\partial f}{\partial X_1} + \ldots + l_{i,n}\frac{\partial f}{\partial X_n} = \sum_{k=1}^{n} l_{i,k} \sum_{j=1}^{r} \frac{\partial y_j}{\partial X_k}\frac{\partial f}{\partial y_j}$$
$$= \sum_{j=1}^{r} \frac{\partial f}{\partial y_j} \sum_{k=1}^{n} l_{i,k}\frac{\partial y_j}{\partial X_k} = \sum_{j=1}^{r} \frac{\partial f}{\partial y_j} L_i(y_j) = 0$$

Thus by definition of the catalecticant matrix, the vectors formed by the coefficients of L_i are in the kernel of $\mathcal{C}_f$. This implies that $\mathrm{rk}(\mathcal{C}_f) \leq r$. □

4.2 Essential variables of higher-order vanishing polynomials

Let $C = \mathrm{Gop}(\boldsymbol{x}, g) \subset \mathbb{F}_2^n$ be a proper irreducible binary Goppa code that is p-distinguishable. Let $\boldsymbol{H} \in \mathbb{F}_2^{rm \times n}$ be the parity check matrix of C in reduced row-echelon form. Let $\boldsymbol{A}$ be the generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ defined in Equation 3, and $\boldsymbol{R} \in \mathbb{F}_{2^m}^{mr \times mr}$ be the invertible matrix such that $\boldsymbol{H} = \boldsymbol{R}\boldsymbol{A}$. Throughout this section, we make the assumption (see Remark 2) that the first s_* columns (in which we shorten) of $\boldsymbol{H}$ are linearly independent, where

$$s_* \coloneqq (e+2)\cdot r - (p-1)\cdot 2^e - p \text{ with } e \coloneqq 1 + \left\lfloor \log_2\left(\frac{r-1}{p-1}\right)\right\rfloor$$

Let $\boldsymbol{H}^{[s_*]}$ be the matrix obtained from $\boldsymbol{H}$ by deleting the first s_* rows. In Theorem 1, we showed that there exist polynomials

$$h_0, \ldots, h_{m-1} \in \ker\left(\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}\right) \subset \mathbb{F}_{2^m}[\boldsymbol{Y}]$$

such that for every ℓ, the polynomial $h_\ell(\boldsymbol{X}\cdot\boldsymbol{R}^T)$ only depends on the indeterminates $X_i^{(u)}$ with $\ell \leq u \leq \ell + e + 1$ and $0 \leq i < r$, where $\boldsymbol{X}$ is the tuple of indeterminates defined in Equation 4. In this section, we use this to derive bounds on the number of essential variables of linear combinations of the polynomials h_ℓ. First, we derive a trivial bound that holds for any linear combination:

Lemma 4. *For any linear combination*

$$f = \lambda_1 \cdot h_{\alpha_1} + \ldots + \lambda_k \cdot h_{\alpha_k} \in \mathbb{F}_{2^m}[\boldsymbol{Y}]$$

where $0 \leq \alpha_j < m$ *and* $\lambda_j \in \mathbb{F}_{2^m}$ *for all* $1 \leq j \leq k$, *we have* $N_{\mathrm{ess}}(f) \leq k \cdot p^2$.

Proof. By Theorem 1, every h_ℓ is a $p \times p$ determinant of a matrix whose entries are linear forms. Hence $N_{\mathrm{ess}}(h_\ell) \leq p^2$ for all ℓ, which implies the claim. □

A key part of our key recovery algorithm will be to compute linear combinations f as in the previous result for which there is some value $0 \leq u < m$ such that $f(\boldsymbol{X}\cdot\boldsymbol{R}^T)$ does not depend on indeterminates $X_i^{(u)}$ for any $0 \leq i < r$. The following uses Theorem 1 to show that, if we take a linear combination of polynomials h_ℓ for a few *subsequent* values of ℓ, we can obtain such polynomials and we can derive a different upper bound on the number of their essential variables.

Table 4. Required degree p and shortening s_* to distinguish a given Goppa code with indicated parameters, as well as the estimated upper bound P_{fail} on the probability that a randomly sampled element from the $\mathbb{F}_{2^m}$-span of the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ (where $\boldsymbol{H}^{[s_*]}$ is a generator matrix of the dual of the Goppa code shortened in s_* positions) with upper bound $(m-1)r-s_*$ on the rank of its first catalecticant is not of the desired form $f_\ell^{[\boldsymbol{\lambda}]}$.

(n, r, m)	$(3488, 64, 12)$	$(495, 8, 9)$	$(482, 7, 9)$	$(254, 8, 8)$	$(221, 7, 8)$	$(120, 5, 7)$
p	22	3	3	4	4	4
s_*	150	21	17	16	12	5
e	2	2	2	2	2	1
$(m-e-2)p^2$	2592	45	45	64	64	64
$(m-1)r-s_*$	534	43	39	40	37	25
P_{fail}	2^{-84}	2^{-25}	2^{-25}	2^{-24}	2^{-24}	2^{-21}

Proposition 11. *For any $0 \leq \ell < m$ and $\boldsymbol{\lambda} \in \mathbb{F}_{2^m}^{m-e-2}$, the linear combination*

$$f_\ell^{[\boldsymbol{\lambda}]} = \lambda_0 \cdot h_\ell + \lambda_1 \cdot h_{\ell+1} + \ldots + \lambda_{m-e-3} \cdot h_{\ell+m-e-3}$$

(where we take the indices of the h_i to be mod m) has the property that $f_\ell^{[\boldsymbol{\lambda}]}(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ only depends on indeterminates $X_i^{(u)}$ where $u \neq \ell - 1 \pmod m$. Furthermore,

$$N_{\text{ess}}(f_\ell^{[\boldsymbol{\lambda}]}) \leq \min((m-e-2) \cdot p^2, (m-1) \cdot r - s_*)$$

Proof. By Theorem 1, $f_\ell^{[\boldsymbol{\lambda}]}(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ only depends on the indeterminates $X_i^{(u)}$ where $u \neq \ell - 1 \pmod m$. Furthermore, due to the shortening, $f_\ell^{[\boldsymbol{\lambda}]}(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ can be expressed in terms of the linear forms in these indeterminates that vanish when evaluated at the s_* columns of $\boldsymbol{A}$ where we shorten. There are $(m-1)r-s_*$ such linear forms, which proves the claim. □

As indicated above, in our key recovery algorithm, we will want to compute elements of the form $f_\ell^{[\boldsymbol{\lambda}]}$ above. We will do this via a brute force search for polynomials whose first catalecticant has rank at most some bound B. Proposition 11 suggests that $B = \min((m-e-2) \cdot p^2, (m-1) \cdot r - s')$ is a reasonable choice of bound. However, one needs to be careful:

– Suppose that

$$(m-e-2) \cdot p^2 \leq (m-1) \cdot r - s'$$

Then *any* linear combination of $e+2$ many of the h_ℓ (and not only those where we have $e+2$ *subsequent* values of ℓ) satisfies this bound. So in this case, with high probability, a brute force search for a polynomial f in the $\mathbb{F}_{2^m}$-span of the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ with $\text{rk}(\mathcal{C}_f) \leq (m-e-2) \cdot p^2$ cannot be expected to yield an element of the form $f_\ell^{[\boldsymbol{\lambda}]}$.

– Suppose that

$$(m-e-2)\cdot p^2 > (m-1)\cdot r - s_*$$

There are at least $m\cdot(2^m-1)^{m-e-2}$ distinct elements of the form $f_\ell^{[\boldsymbol{\lambda}]}$. We will approximate this by $m\cdot 2^{m(m-e-2)}$. On the other hand, we assume that linear combinations of more than $N \coloneqq \lfloor \frac{(m-1)\cdot r - s_*}{p^2} \rfloor$ of the h_ℓ will lead to a higher rank of the first catalecticant than $(m-1)\cdot r - s_*$, unless this linear combination is of the form $f_\ell^{[\boldsymbol{\lambda}]}$. In total, there are at most $\binom{m}{N}\cdot 2^{m\cdot N}$ linear combinations of N many of the polynomials h_ℓ. Thus, if we do a brute force search for a polynomial f in the $\mathbb{F}_{2^m}$-span of the kernel of $\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}$ subject to $\mathrm{rk}(\mathcal{C}_f) \leq (m-1)\cdot r - s_*$, then the probability that we do not obtain an element of the form $f_\ell^{[\boldsymbol{\lambda}]}$ is bounded above by

$$P_{\text{fail}} \coloneqq \frac{\binom{m}{N}\cdot 2^{mN}}{m\cdot 2^{m(m-e-2)}}$$

Due to the condition that $(m-e-2)\cdot p^2 > (m-1)\cdot r - s_*$, we have $N < m-e-2$, so for most parameter sets, P_{fail} is quite small. In Table 4, we tabulate this probability for some parameter sets that occur in the TII challenges that we broke as well as for the smallest Classic McEliece parameter set.

The above considerations lead us to the following heuristic assumption:

Heuristic Assumption 2. Suppose $C = \mathrm{Gop}(\boldsymbol{x}, g) \subset \mathbb{F}_2^n$ is a proper irreducible binary Goppa code that is p-distinguishable. Let $\boldsymbol{H} \in \mathbb{F}_2^{rm\times n}$ be the parity check matrix of C in reduced row-echelon form. Let $\boldsymbol{A}$ be the generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ defined in Equation 3 and let $\boldsymbol{R} \in \mathbb{F}_{2^m}^{rm\times rm}$ be the invertible matrix such that $\boldsymbol{H} = \boldsymbol{R}\boldsymbol{A}$. Suppose that the first s_* columns of $\boldsymbol{H}$ are linearly independent, where s_* is as defined in Equation 8. Let $\boldsymbol{H}^{[s_*]}$ be the matrix obtained from $\boldsymbol{H}$ by deleting the first s_* rows. Suppose further that

$$(m-e-2)\cdot p^2 > (m-1)\cdot r - s_* \tag{9}$$

Then with high probability, a randomly selected element

$$f \xleftarrow{\$} \left\{ h \in \mathrm{span}_{\mathbb{F}_{2^m}}\left(\ker\left(\tilde{\varphi}^{(p)}_{\boldsymbol{H}^{[s_*]}}\right)\right) \mid \mathrm{rk}(\mathcal{C}_h) \leq (m-1)\cdot r - s_* \right\}$$

has the property that for some $0 \leq u < m$, the polynomial $f(\boldsymbol{X}\cdot\boldsymbol{R}^T)$ does not depend on any indeterminate $X_i^{(u)}$ for $0 \leq i < r$.

We have verified this assumption in many experiments, including our key recovery attacks against TII challenges. We have also verified in examples that if the condition in Equation 9 is violated, then a brute force search often does find elements which are not of the desired form.

Algorithm 3 Recovery of support and Goppa polynomial from a parity check matrix of a binary Goppa code

Input: $r, m, n \in \mathbb{N}$, $\boldsymbol{H} \in \mathbb{F}_2^{rm \times n}$
Output: $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ and $g \in \mathbb{F}_{2^m}[X]$ with $\deg(g) = r$ such that $\mathrm{Gop}(\boldsymbol{x}, g)$ has parity check matrix $\boldsymbol{H}$

$U_{-1} \leftarrow \emptyset$; $U_0 \leftarrow \{\boldsymbol{0}\} \subset \mathbb{F}_2^n$
$i \leftarrow 0$
while $\dim(U_i) \neq \dim(U_{i-1})$ **do**
    $i \leftarrow i + 1$
    $W \leftarrow$ ComputeBlockOmittingSubspace$(r, m, n, \boldsymbol{H})$ ▷ Algorithm 4
    Determine $0 \leq j_0 < m$ such that $\dim(U + W^{(2^j)})$ is minimal for $j = j_0$.
    Set $U_i \leftarrow U_{i-1} + W^{(2^{j_0})}$.
end while
$U \leftarrow U_i$
for $j = 1, \ldots, m-1$ **do**
    $U \leftarrow U \cap U^{(2)}$
    if $\dim(U) < 2r$ **then**
        $U = U * U$
        $\boldsymbol{x}, \boldsymbol{y} \leftarrow$ ShestakovSidelnikov(U)
        $(\boldsymbol{x}', g) \leftarrow$ Algorithm 2 applied to $(\boldsymbol{x}, \boldsymbol{H})$
        return $(\boldsymbol{x}, g)$
    end if
end for

5 Key recovery attack

Suppose $C = \mathrm{Gop}(\boldsymbol{x}, g)$ is a proper binary Goppa code with support vector $\boldsymbol{x} \in \mathbb{F}_{2^m}^n$ and irreducible Goppa polynomial $g \in \mathbb{F}_{2^m}[X]$ of degree r. Let $\boldsymbol{H} \coloneqq \mathbb{F}_2^{mr \times n}$ be a parity check matrix of C in reduced row-echelon form. Our aim is to recover a valid support and Goppa polynomial for the underlying Goppa code from $\boldsymbol{H}$.

5.1 Outline of the attack

The full key recovery algorithm is described in Algorithm 3. In the following, we explain the algrithm and individual steps in detail.

By Proposition 2, extension of scalars to $\mathbb{F}_{2^m}$ leads to

$$\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp) = \sum_{i=0}^{m-1} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

where $\boldsymbol{y} = \frac{1}{g(\boldsymbol{x})}$. Since we assumed the underlying Goppa code to be proper, the sum is direct.

Our aim is to retrieve a basis of one of the GRS codes $\mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$ contained in this space. Once we achieve this, we can apply the Shestakov-Sidelnikov attack [22] to recover a valid support and multiplier (and then the Goppa polynomial) of the Goppa code. Note that it is actually sufficient to compute a basis

of one of the subspaces

$$V_\ell \coloneqq \sum_{0 \le i \le m-1, i \neq \ell} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

for some $0 \le \ell < m$. This is because for any $0 \le \ell < m$, we have

$$V_\ell \cap V_\ell^{(2)} \cap \ldots \cap V_\ell^{(2^{m-2})} = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^{\ell-1})}$$

For convenience, we refer to the elements of V_ℓ as *ℓ-block omitting* elements. Finding elements in one of the V_ℓ constitutes the core of our algorithm and we describe how to do this in subsection 5.3. For now, we assume that we have a (probabilistic) algorithm ComputeBlockOmittingSubspace() at our disposal that can sample a subspace W which is guaranteed to be contained in some V_ℓ. We will see that in any single run of this algorithm, we expect the subspace W which is output to have dimension about $(m-1)r - s_*$ where s_* is as defined in Equation 7. However, we cannot assume to know *which* V_ℓ it is contained in.

In lines 1-9 of Algorithm 3, we use ComputeBlockOmittingSubspace() to construct a large subspace U of one of the V_ℓ. In essence, we keep sampling elements (or rather smaller subspaces) of block-omitting elements and add them to our space U in each iteration until we cannot sample any new elements that increase the dimension of U. However, one complication that we have to deal with is that once we have sampled a block-omitting subspace W, we do not know which V_ℓ it is contained in and whether it is contained in the same one as the subspace U_{i-1} we have obtained so far. Therefore, in line 6 of the algorithm, we consider all Frobenius powers $W^{(2^j)}$ of W. We expect it to lie in a common block-omitting subspace with U_{i-1} if the direct sum of both spaces has minimal dimension. Then we extend the space U_{i-1} accordingly in line 7.

In almost all experiments that we have run, the subspace U that we obtain in this way has dimension $(m-1)r$ and is then indeed equal to one of the block-omitting subspaces. An exception were the TII challenges with parameter set $m = 8$ and $r = 7$. For these, we could only obtain a subspace of codimension 3 of one of the block-omitting subspaces. We will explain in subsection 5.2 how to deal with this case.

For now, suppose that U is one of the block-omitting subspaces. Then we can proceed as indicated above: We compute the intersection $U \cap U^{(2)} \cap U^{(4)} \cap \ldots$ until the resulting space has dimension r. Then we have obtained the basis of one of the GRS codes and can obtain the support and multiplier using the Shestakov-Sidelnikov attack.

Note that Algorithm 3 is actually described slightly differently: When, after a suitable number of intersections of Frobenius conjugates, we have obtained a space $U = \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{2^i}$ of dimension r, it computes its Schur square $U * U = GRS_{2r-1}(\boldsymbol{x}, \boldsymbol{y}^2)^{2^i}$ and applies the Shestakov-Sidelnikov attack to it. This still recovers a valid support $\boldsymbol{x}$ and allows to compute a suitable Goppa polynomial using Algorithm 2. We choose to do it this way because this also covers the exceptional case when we cannot obtain the full block-omitting subspace but

only a subspace of it of small codimension, as we shall see in the following section.

5.2 Dealing with large proper subspaces of V_ℓ

Suppose that, as explained above, in line 9 we obtain a large but proper subspace U of V_ℓ. The idea on how to deal with this case is already explained in [2], but we reproduce it here for convenience. Since $U \subset V_\ell$, we know that

$$U' \coloneqq U \cap U^{(2)} \cap \ldots \cap U^{(2^{m-3})} \subset \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^{\ell-2})} + \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^{\ell-1})}$$

By Lemma 1, we have

$$\mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^{\ell-2})} + \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^{\ell-1})} = \mathrm{GRS}_{2r}(\boldsymbol{x}, \boldsymbol{y}^2)^{(2^{\ell-2})}$$

Taking the Schur square, we deduce that

$$U' * U' \subset \mathrm{GRS}_{4r-1}(\boldsymbol{x}, \boldsymbol{y}^4)^{(2^{\ell-2})}$$

One may hope that U' is large enough that equality holds. Indeed, this was the case in all the experiments we have conducted. Assuming this, we can recover a valid support $\boldsymbol{x}'$ and multiplier for $\mathrm{GRS}_{4r-1}(\boldsymbol{x}, \boldsymbol{y}^4)^{(2^{\ell-2})}$ using the Shestakov-Sidelnikov attack. Note that $\boldsymbol{x}'$ is then a valid support for the Goppa code as an alternant code, but the multiplier is not a valid multiplier for the Goppa code. But we can use Algorithm 2 to recover a valid Goppa support and Goppa polynomial from $\boldsymbol{x}'$ and the parity check matrix $\boldsymbol{H}$.

5.3 Computation of block-omitting elements

The algorithm is described in Algorithm 4.

First, using Algorithm 5, we compute a degree p and a shortening s such that Goppa codes with the given parameters can be distinguished by applying the higher-order vanishing distinguisher at degree p to the dual code shortened in s positions. Algorithm 5 also ensures that the condition in Equation 9 from Heuristic Assumption 2 is satisfied.

In lines 2-5, the columns of the parity check matrix $\boldsymbol{H}$ of the binary Goppa code are randomly permuted, then the matrix is echelonised and we apply shortening in the first s positions by deleting the first s rows of the echelonised matrix. Note that we assume here that the first s columns are linearly independent, but this turns out to be the case with high probability (see Remark 2). Applying a random permutation of the columns beforehand results in a random choice of positions in which we shorten.

In line 6, we compute a basis of the space of higher-order vanishing polynomials of degree p. Note that by Heuristic Assumption 1, we expect to obtain exactly m polynomials in the basis. Indeed, this has always been the case in our experiments. Note that up to this point, we can conduct all computations over the binary field $\mathbb{F}_2$.

Algorithm 4 Compute a block-omitting subspace

Input: $r, m, n \in \mathbb{N}$; parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{rm \times n}$ of binary Goppa code with extension degree m, degree r and length n

Output: A set $U \subset \mathbb{F}_{2^m}^n$ of block-omitting vectors

$(p, s) \leftarrow$ DistinguishingParameters(r, m, n) ▷ Algorithm 5
Randomly sample a permutation matrix $\boldsymbol{P} \in \mathbb{F}_2^{n \times n}$
$\boldsymbol{H}' \leftarrow \boldsymbol{H} \cdot \boldsymbol{P}$ ▷ randomly permute columns
$\boldsymbol{H}'' \leftarrow$ ReducedRowEchelonForm$(\boldsymbol{H}')$
$\boldsymbol{H}''' \leftarrow$ matrix obtained by deleting the first s rows of $\boldsymbol{H}''$ ▷ shortening
$\{v_1, \ldots, v_m\} \leftarrow$ basis of $\ker\left(\tilde{\varphi}^{(p)}_{\boldsymbol{H}'''}\right) \subset \mathbb{F}_2[Y_{s+1}, \ldots, Y_{mr}]$
for $j = 1, \ldots, m$ **do**
    $\boldsymbol{M}_j \leftarrow$ Catalecticant(v_j) ▷ See Definition 8.
end for
Compute $\lambda_1, \ldots, \lambda_m \in \mathbb{F}_{2^m}$ such that $\mathrm{rk}(\lambda_1 \boldsymbol{M}_1 + \ldots, \lambda_m \boldsymbol{M}_m) \leq (m-1)r - s$ via brute force search.
$v \leftarrow \lambda_1 v_1 + \ldots + \lambda_m v_m$
$U \leftarrow \emptyset$
while $\dim_{\mathbb{F}_{2^m}}(\mathrm{span}_{\mathbb{F}_{2^m}}(U)) < \mathrm{rk}(\mathcal{C}_v)$ **do**
    Randomly choose distinct $i_1, \ldots, i_{p-1} \in \{s+1, s+2, \ldots, mr\}$.
    $\tilde{v} \leftarrow \frac{\partial^{p-1} v}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}}$
    $\boldsymbol{u} \leftarrow$ coefficients of $Y_{s+1}, \ldots, Y_{mr}$ in $\tilde{v}$, written as a vector in $\mathbb{F}_{2^m}^{mr-s}$
    $U \leftarrow U \cup \{\boldsymbol{u} \cdot \boldsymbol{H}''' \cdot \boldsymbol{P}^{-1}\}$
end while
return U

Algorithm 5 Compute parameters that allow to distinguish Goppa codes

Input: $r, m, n \in \mathbb{N}$

Output: $p, s \in \mathbb{N}$ such that binary Goppa codes of degree r, extension degree m and length n are p-distinguishable using the higher-order vanishing distinguisher after shortening in s positions

$p = 3$
while True **do**
    $e \leftarrow 1 + \lfloor \log_2((r-1)/(p-1)) \rfloor$
    $s \leftarrow (e+2) \cdot r - (p-1) \cdot 2^e - p$
    if $s < 0$ **then**
        return $\perp$ ▷ Code cannot be distinguished
    end if
    $c \leftarrow$ degree p coefficient of $\left[(1+t)^{mr-s}\left(1-t^2\right)^{n-mr}\right]^+$
    if $c = 0$ and $(m-e-2) \cdot p^2 > (m-1) \cdot r - s$ **then**
        return (p, s)
    else
        $p \leftarrow p + 1$
    end if
end while

By Theorem 1, we know that there are m polynomials

$$h_0, \ldots, h_{m-1} \in \mathrm{span}_{\mathbb{F}_{2^m}}\{v_1, \ldots, v_m\}$$

satisfying the properties in Theorem 1. Using a brute force search, in lines 7-11 we compute a polynomial v in the linear span of the v_ℓ such that its first catalecticant has rank at most $(m-1)r - s$. We will analyse the complexity of this brute force search in subsection 5.4. Assuming that Heuristic Assumption 2 holds, with high probability, there is $0 \leq u < m$ such that $v(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ does not depend on any of the indeterminates $X_i^{(u)}$. We want to show how we can obtain block-omitting vectors from such a polynomial. First, we prove a preparatory technical lemma.

Lemma 5. *Suppose $f = \lambda_1 Y_1 + \ldots + \lambda_k Y_k$ is a linear form in the polynomial ring $\mathbb{F}_q[Y_1, \ldots, Y_k]$ and let $\boldsymbol{S} \in \mathbb{F}_q^{k \times k}$ be such that $f(\boldsymbol{X} \cdot \boldsymbol{S}^T)$ depends only on indeterminates X_i for $i \in I \subset \{1, \ldots, k\}$. Let $\boldsymbol{C} \in \mathbb{F}_q^{k \times n}$ and $\boldsymbol{C'} = \boldsymbol{SC}$ and denote by $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k \in \mathbb{F}_q^n$ the rows of $\boldsymbol{C}$ and by $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k \in \mathbb{F}_q^n$ the rows of $\boldsymbol{C'}$. Then*

$$\lambda_1 \cdot \boldsymbol{c}'_1 + \ldots + \lambda_k \cdot \boldsymbol{c}'_k \in \mathrm{span}_{\mathbb{F}_q}\{\boldsymbol{c}_i \mid i \in I\}$$

Proof. We denote the rows of $\boldsymbol{S}$ by $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k \in \mathbb{F}_q^k$ and the entries of $\boldsymbol{S}$ by $s_{i,j}$ where i is the row index and j the column index. Then we have

$$\sum_{i=1}^{k} \lambda_i \cdot \boldsymbol{c}'_i = \sum_{i=1}^{k} \lambda_i \cdot \boldsymbol{s}_i \cdot \boldsymbol{C} = \sum_{i=1}^{k} \lambda_i \cdot \sum_{j=1}^{k} s_{i,j} \boldsymbol{c}_j \tag{10}$$

On the other hand, we have

$$f(\boldsymbol{X} \cdot \boldsymbol{S}^T) = \sum_{i=1}^{k} \lambda_i \cdot \sum_{j=1}^{k} X_j \cdot s_{i,j} \tag{11}$$

By assumption, this expression only depends on X_i for $i \in I$. The expression in Equation 10 is obtained from Equation 11 by substituting $X_i \mapsto \boldsymbol{c}_i$, so it only depends on $\boldsymbol{c}_i$ for $i \in I$ as claimed. □

The following proposition shows that given a polynomial v as computed in line 11 of Algorithm 4, the procedure in lines 14-17 computes vectors in a block-omitting subspace.

Proposition 12. *Let $C = \mathrm{Gop}(\boldsymbol{x}, g)$ be a proper irreducible binary Goppa code. Let $\boldsymbol{H} \in \mathbb{F}_2^{mr \times n}$ be the parity check matrix of C in reduced row-echelon form, let $\boldsymbol{A}$ be the generator matrix of $\mathrm{span}_{\mathbb{F}_{2^m}}(C^\perp)$ defined in Equation 3 and let $\boldsymbol{R} \in \mathbb{F}_{2^m}^{rm \times rm}$ be the invertible matrix such that $\boldsymbol{H} = \boldsymbol{RA}$.*

Suppose $v(\boldsymbol{Y}) \in \mathbb{F}_{2^m}[\boldsymbol{Y}]$ is a polynomial such that there is $0 \leq u < m$ such that $v(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ does not depend on any of the indeterminates $X_i^{(u)}$ for $0 \leq i < r$. Let

$$L(Y_1, \ldots, Y_{mr}) \coloneqq \frac{\partial^{p-1} v}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}} = \lambda_1 Y_1 + \ldots + \lambda_{mr} Y_{mr}$$

be some partial derivative of v of order $p-1$ where $1 \leq i_1, \ldots, i_{p-1} \leq mr$ are arbitrary. Then

$$\lambda_1 \boldsymbol{h}_1 + \ldots + \lambda_{mr} \boldsymbol{h}_{mr} \in \sum_{i \neq u} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{(2^i)}$$

where $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{mr}$ denote the row vectors of $\boldsymbol{H}$.

Proof. Since v has the property that $v(\boldsymbol{X}\boldsymbol{R}^T)$ does not depend on the indeterminates $X_i^{(u)}$ for $0 \leq i < r$, by the chain rule for differentiation, the same holds for any partial derivative of v. In particular, L has this property. The rows of the matrix $\boldsymbol{A}$ corresponding to indeterminates $X_i^{(j)}$ for $j \neq u$ all lie in $\sum_{i \neq u} \mathrm{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{2^i}$ by definition of $\boldsymbol{A}$. The same then holds for $\lambda_1 \boldsymbol{h}_1 + \ldots + \lambda_{mr} \boldsymbol{h}_{mr}$ by Lemma 5. □

Thus, assuming that Heuristic Assumption 2 holds, with high probability, the polynomial v computed in line 11 of Algorithm 4 has the property that there is $0 \leq u < m$ such that $v(\boldsymbol{X} \cdot \boldsymbol{R}^T)$ does not depend on any of the indeterminates $X_i^{(u)}$. Thus, by Proposition 12, the steps in lines 14-17 really do output vectors in a block-omitting subspace. We can deduce the number of linearly independent block-omitting vectors we can obtain from one polynomial v using the following lemma.

Lemma 6. *Let $g \in \mathbb{F}_{2^m}[Y_1, \ldots, Y_k]^{(p)}$ be a homogeneous polynomial of degree p such that every indeterminate Y_i occurs to degree at most 1, and let*

$$U \coloneqq \mathrm{span}_{\mathbb{F}_{2^m}} \left\{ \frac{\partial^{p-1} g}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}} \mid 1 \leq i_1 < \ldots < i_{p-1} \leq k \right\} \subset \mathbb{F}_{2^m}[Y_1, \ldots, Y_k]^{(1)}$$

be the space spanned by all linear forms which are obtained by computing partial derivatives of g of order $p-1$. Then $\dim_{\mathbb{F}_{2^m}}(U) = \mathrm{rk}(\mathcal{C}_g)$, the rank of the first catalecticant of g.

Proof. Let T_1 be the $\mathbb{F}_{2^m}$-vector space with a basis $\partial_1, \ldots, \partial_k$ (which we shall interpret as the partial differential operators). Denote by $S_1 = \mathbb{F}_{2^m}[Y_1, \ldots, Y_k]^{(1)}$ the space of linear forms in the given indeterminates. Define the bilinear pairing

$$T_1 \times S_1 \to \mathbb{F}_{2^m}, \quad (a_1\partial_1 + \ldots + a_k\partial_k, f) \mapsto a_1 \frac{\partial f}{\partial Y_1} + \ldots + a_k \frac{\partial f}{\partial Y_k}$$

This pairing is clearly non-degenerate. Let $U^\perp \subset T_1$ be the dual of U under this pairing. By non-degeneracy, $\dim(U) = k - \dim(U^\perp)$. We claim that

$$U^\perp = \{\partial \in T_1 \mid \partial g = 0\}$$

If this holds, then $\dim(\ker(\mathcal{C}_g)) = \dim(U^\perp)$ and hence the claim follows since then we have $\mathrm{rk}(\mathcal{C}_g) = k - \dim(U^\perp) = \dim(U)$.

The inclusion $\{\partial \in T_1 \mid \partial g = 0\} \subset U^\perp$ is clear from the definition of U. For the reverse inclusion, let $\partial \in U^\perp$ be an arbitrary element. Suppose $\partial g \neq 0$. Then there is a monomial $Y_{i_1} \cdots Y_{i_{p-1}}$ in ∂g that has a coefficient $a \neq 0$. But then

$$0 \neq a = \frac{\partial^{p-1}(\partial g)}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}} = \partial \left(\frac{\partial^{p-1} g}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}} \right)$$

where for the first equality we used that each indeterminate occurs to degree at most 1. Since $\frac{\partial^{p-1} g}{\partial Y_{i_1} \cdots \partial Y_{i_{p-1}}} \in U$, this contradicts the assumption that $\partial \in U^\perp$. This concludes the proof. □

As a consequence of this lemma, we see that the dimension of the space U defined in line 14 of Algorithm 4 is $\mathrm{rk}(\mathcal{C}_v)$. This is why, in lines 13-18, we can stop computing block-omitting vectors once U spans a subspace of dimension $\mathrm{rk}(\mathcal{C}_v)$. In most of our experiments breaking TII challenges, $\mathrm{rk}(\mathcal{C}_v)$ actually attained its upper bound $(m-1)r - s$. The only exception was the TII challenges with parameters $m = 8, r = 7$ where $\mathrm{rk}(\mathcal{C}_v)$ was slightly lower than this bound. This shows that one call of Algorithm 4 already computes a large space of block-omitting elements and that the loop starting in line 3 of Algorithm 3 only needs to be run a few times before a large enough subspace of block-omitting elements is obtained. In our TII experiments, 2 or 3 iterations always sufficed for a successful key recovery.

5.4 Complexity

The bottleneck of Algorithm 3 is the computation of block-omitting subspaces via Algorithm 4. All other steps run in polynomial time. Within Algorithm 4, there are two steps which potentially dominate the complexity of the overall algorithm:

- The computation of higher-order vanishing polynomials in line 6 entails the computation of the kernel of a large matrix. [13, Proposition 10] shows that the matrices are sparse so that Wiedemann's algorithm [23] can be used, and provides a complexity estimate. For Classic McEliece parameters, the dominant term of this complexity estimate (ignoring lower-order terms as well as constant factors) is given as κ_W in Table 2.
- In line 10, we essentially solve a minrank problem via a brute-force search. In subsection 4.2, we saw that there are at least $m \cdot (2^m - 1)^{m-e-2} \approx m \cdot 2^{m(m-e-2)}$ solutions, and it suffices to find one of them. Since the search space has a total size of 2^{m^2} matrices, we expect to have to compute the rank of about $\frac{1}{m} \cdot 2^{m(e+2)}$ matrices, each having $mr - s$ rows, until we find a solution. Roughly bounding the complexity of the rank computation of such a matrix by $(mr-s)^3$, we expect a run time of about $\frac{(mr-s)^3}{m} \cdot 2^{m(e+2)}$.

These observations yield the following result.

Proposition 13. *With* $p = \textsc{DistinguishingDegree}(r, m, n)$ *and* $e(r, p) := e = 1 + \lfloor \log_2((r-1)/(p-1)) \rfloor$ *and* $s = (e+2) \cdot r - (p-1) \cdot 2^e - p$*, the run time of Algorithm 3 is*

$$\mathcal{O}\left(c_{\mathrm{BlockOm}}(m, r, p) + c_{\mathrm{Wied}}(m, r, n, p)\right)$$

where

$$c_{\mathrm{BlockOm}}(m, r, p, s) := \frac{(mr-s)^3}{m} \cdot 2^{m(2+e(r,p))}$$

$$c_{\mathrm{Wied}}(m, r, n, p) := N_1 \cdot \left(N_1(n-mr) \sum_{i=1}^{\lfloor \log_2 p \rfloor} \binom{p}{2^i} + N_0 \cdot \log(N_0) \right)$$

with $N_0 := (n-mr) \cdot \sum_{i=1}^{\ell} \binom{mr-s}{p-2^i}$ *and* $N_1 := \binom{mr-s}{p}$.

Proof. This follows from the observations above and the fact from [13, Proposition 10] that c_{Wied} is the order of complexity of the application of Wiedemann's algorithm to the matrix representing $\tilde{\varphi}^{(p)}_{\boldsymbol{H}'''}$. □

6 Application to TII challenges and Classic McEliece

In Table 1, we already provided some details of the TII challenges that we have broken using our approach. In subsection 6.1, we make a few more remarks on this, and in subsection 6.2 we analyse the impact of our key recovery algorithm on Classic McEliece parameters.

6.1 Breaking TII challenges

Using our key recovery algorithm implemented in SageMath and run on a machine equipped with two AMD EPYC 9745 128-Core Processors, we have broken the TII Challenges listed in Table 1 in the Introduction. The secret keys that we obtained as well as a SageMath script to verify their correctness can be found on https://github.com/tobhem/key-recovery-mceliece-tii-solutions. We make some remarks and observations.

Estimated security level. The security level λ_{TII} estimated by the TII was obtained by estimating the complexity of a brute-force search for the Goppa polynomial combined with the support-splitting algorithm to determine a valid support vector. In order to apply the support-splitting algorithm, at least the unordered set of entries of the support vector has to be known. Therefore, the choice of this set needs to be brute-forced if the length n of the Goppa code is smaller than the field size. For the TII challenges with the same values of m and r but different choices of n (such as TII-220 and TII-92), this can lead to vastly different estimates of the security level. In contrast, our key recovery algorithm essentially has the same complexity for the same values of m and r, irrespective of the length n, as long as the same degree p can be used to apply the higher-order vanishing distinguisher.

Table 5. Estimate of the running time of Algorithm 3 for Classic McEliece parameters according to Proposition 13.

(n, r, m)	$(3488, 64, 12)$	$(4608, 96, 13)$	$(6688, 128, 13)$	$(6960, 119, 13)$	$(8192, 128, 13)$
p	22	46	52	41	39
s	150	158	256	275	321
e	2	2	2	2	2
c_{BlockOm}	2^{73}	2^{79}	2^{80}	2^{79}	2^{80}
c_{Wied}	2^{297}	2^{594}	2^{690}	2^{564}	2^{550}

Running time. For Goppa code parameters that are distinguishable for small values of p (say $p = 3$), the brute-force search for block-omitting subspaces (line 10 in Algorithm 4) dominates the complexity and takes much longer than the computation of degree p higher-order vanishing polynomials (line 6 in Algorithm 4). This is the case for TII challenges with $m = 9, r = 7$ and $m = 9, r = 8$ for example. As a result, we see some fluctuation between the run time of the key recovery for TII challenges with the same values of m and r: This is because the brute-force search sometimes terminates more quickly than other times. However, as p increases, the computation of higher-order vanishing polynomial quickly dominates the run time of the whole key recovery, see also subsection 6.2 for Classic McEliece parameters.

TII Challenges not yet broken in practice. Most of the TII Challenges that we have not broken yet in practice require degree $p \geq 5$ in order for the Goppa codes to be distinguishable using the higher-order vanishing distinguisher. This significantly increases the complexity of the computation of the kernel of $\tilde{\varphi}^{(p)}$. For example, TII Challenge 109 has parameters $m = 8, r = 10$ and requires $p = 6$. This results in computing the kernel of a $2^{26.4} \times 2^{31}$ matrix, which is significantly larger than the matrices we had to deal with for the TII challenges that we broke.

6.2 Relevance for Classic McEliece

Assuming that Heuristic Assumption 1 and Heuristic Assumption 2 also hold for Classic McEliece parameters, we conjecture that Algorithm 3 also yields a successful key recovery.

The values relevant to the complexity of Algorithm 3 according to Proposition 13 for Classic McEliece parameters can be found in Table 5. We see that for all parameter sets, the complexity of the computation of higher-order vanishing polynomials is clearly the dominant term. In comparison, the contribution from the brute-force search of block-omitting elements is negligible. For this reason, we conjecture that our approach allows for key recovery for Classic McEliece parameter sets with roughly the complexity with which these parameter sets can be distinguished from random linear codes using the higher-order vanishing distinguisher, i.e. the term given by c_{Wied} in Table 5.

References

1. Alagic, G., Bros, M., Ciadoux, P., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., et al.: Status report on the fourth round of the NIST post-quantum cryptography standardization process. US Department of Commerce, National Institute of Standards and Technology (2025)
2. Bardet, M., Lemoine, A., Tillich, J.: An attack on the CFS scheme and on TII McEliece challenges. IACR Cryptol. ePrint Arch. **2026**, 430 (2026), https://eprint.iacr.org/2026/430
3. Bardet, M., Mora, R., Tillich, J.: Polynomial Time Key-Recovery Attack on High Rate Random Alternant Codes. IEEE Trans. Inf. Theory **70**(6), 4492–4511 (2024), https://doi.org/10.1109/TIT.2023.3334592
4. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theory **24**(3), 384–386 (1978), https://doi.org/10.1109/TIT.1978.1055873
5. Briaud, P., Lemoine, A., Randriambololona, H., Tillich, J.P.: A Heuristic Subexponential Attack on the McEliece Cryptosystem. Cryptology ePrint Archive, Paper 2026/1232 (2026), https://eprint.iacr.org/2026/1232
6. Carlini, E.: Reducing the number of variables of a polynomial. In: Elkadi, M., Mourrain, B., Piene, R. (eds.) Algebraic Geometry and Geometric Modeling, pp. 237–247. Springer (2006), https://doi.org/10.1007/978-3-540-33275-6_15
7. Couvreur, A., Corbella, I.M., Pellikaan, R.: Cryptanalysis of McEliece Cryptosystem Based on Algebraic Geometry Codes and Their Subcodes. IEEE Trans. Inf. Theory **63**(8), 5404–5418 (2017), https://doi.org/10.1109/TIT.2017.2712636
8. Couvreur, A., Mora, R., Tillich, J.: A New Approach Based on Quadratic Forms to Attack the McEliece Cryptosystem. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14441, pp. 3–38. Springer (2023), https://doi.org/10.1007/978-981-99-8730-6_1
9. Couvreur, A., Otmani, A., Tillich, J.: Polynomial Time Attack on Wild McEliece over Quadratic Extensions. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8441, pp. 17–39. Springer (2014), https://doi.org/10.1007/978-3-642-55220-5_2
10. Dür, A.: The automorphism groups of Reed-Solomon codes. Journal of Combinatorial Theory, Series A **44**(1), 69–82 (1987)
11. Faugère, J., Gauthier-Umaña, V., Otmani, A., Perret, L., Tillich, J.: A distinguisher for high rate McEliece cryptosystems. In: 2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011. pp. 282–286. IEEE (2011), https://doi.org/10.1109/ITW.2011.6089437
12. Faugère, J., Otmani, A., Perret, L., Tillich, J.: Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In: Gilbert, H. (ed.) Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6110, pp. 279–298. Springer (2010), https://doi.org/10.1007/978-3-642-13190-5_14
13. Hemmert, T., Wiemers, A.: Distinguishing Goppa codes using higher-order vanishing. Cryptology ePrint Archive, Paper 2025/1661 (2025), https://eprint.iacr.org/2025/1661

14. International Organization for Standardization, International Electrotechnical Commission: Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers — Amendment 2. Standard ISO/IEC 18033-2:2006/Amd 2:2026, International Organization for Standardization, Geneva, Switzerland (2026)
15. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1992), seventh impression
16. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report **42-44**, 114–116 (1978)
17. Mora, R.: On the matrix code of quadratic relationships for a Goppa code. Adv. Math. Commun. **19**(3), 829–852 (2025), https://doi.org/10.3934/amc.2024026
18. Panny, L.: On Breaking McEliece Keys Using Brute Force. In: Bardet, M., Niederhagen, R. (eds.) Post-Quantum Cryptography - 17th International Workshop, PQCrypto 2026, Saint-Malo, France, April 14-16, 2026, Proceedings, Part II. pp. 275–308. Lecture Notes in Computer Science, Springer (2026), https://doi.org/10.1007/978-3-032-22698-3_9
19. Pellikaan, R., Márquez-Corbella, I.: Error-correcting pairs for a public-key cryptosystem. In: Journal of Physics: Conference Series. vol. 855, p. 012032. IOP Publishing (2017)
20. Randriambololona, H.: The syzygy distinguisher. IACR Cryptol. ePrint Arch. p. 1193 (16/10/2025), https://eprint.iacr.org/2024/1193
21. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Trans. Inf. Theory **46**(4), 1193–1203 (2000), https://doi.org/10.1109/18.850662
22. Sidelnikov, V.M., Shestakov, S.O.: On insecurity of cryptosystems based on generalized Reed-Solomon codes. Discrete Mathematics and Applications **2**(4), 439–444 (1992), https://doi.org/10.1515/dma.1992.2.4.439
23. Wiedemann, D.H.: Solving sparse linear equations over finite fields. IEEE Trans. Inf. Theory **32**(1), 54–62 (1986), https://doi.org/10.1109/TIT.1986.1057137
24. Wieschebrink, C.: Cryptanalysis of the Niederreiter Public Key Scheme Based on GRS Subcodes. In: Sendrier, N. (ed.) Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6061, pp. 61–72. Springer (2010), https://doi.org/10.1007/978-3-642-12929-2_5