On Generalized Reed-Solomon Codes Over Commutative and Noncommutative Rings

Guillaume Quintin, Morgan Barbier, Christophe Chabot

On Generalized Reed-Solomon Codes Over Commutative and Noncommutative Rings

Guillaume Quintin, Morgan Barbier, Christophe Chabot

***Abstract*—In this paper we study generalized Reed-Solomon codes (GRS codes) over commutative, noncommutative rings, show that the classical Welch-Berlekamp and Guruswami-Sudan decoding algorithms still hold in this context and we investigate their complexities. Under some hypothesis, the study of noncommutative generalized Reed-Solomon codes over finite rings leads to the fact that GRS code over commutative rings have better parameters than their noncommutative counterparts. Also GRS codes over finite fields have better parameters than their commutative rings counterparts. But we also show that given a unique decoding algorithm for a GRS code over a finite field, there exists a unique decoding algorithm for a GRS code over a truncated power series ring with a better asymptotic complexity. Moreover we generalize a lifting decoding scheme to obtain new unique and list decoding algorithms designed to work when the base ring is for example a Galois ring or a truncated power series ring or the ring of square matrices over the latter ring.**

***Index Terms*—Algebra, Algorithm design and analysis, Decoding, Error correction, Reed-Solomon codes.**

I. Introduction

Reed-Solomon codes (denoted by RS codes in the rest of this paper) form an important and well-studied family of codes. They were first proposed in 1960 by Irvin Stoy Reed and Gustave Solomon in their original paper [36]. They have the property to be *Maximum Distance Separable* (MDS) codes, thus reaching the Singleton bound. Not only do RS codes have the best possible parameters, they also can be efficiently decoded. See for example [22] and [27]. They are widely used in practice such as in compact disc players, disk drives, satellite communications, and high-speed modems such as ADSL. See [43] for details about applications of RS codes. A breakthrough has been made by Madhu Sudan in 1997 about the list decoding of RS codes in his paper [41], further improved by Venkatesan Guruswami and Madhu Sudan in [26]. They showed that RS codes are list decodable up to the generic Johnson bound in polynomial time. In [32], Rasmus Refslund Nielsen and Tom Høholdt showed that the probability of having more than one codeword returned by any list decoding algorithm which can decode up to the generic Johnson bound is very small, making the Guruswami-Sudan algorithm usable in practice.

Guillaume Quintin is with the Laboratoire d'informatique de l'X (LIX), École polytechnique, 91128 Palaiseau, FRANCE, UMR 7161 X-CNRS, e-mail: quintin@lix.polytechnique.fr.

Morgan Barbier is with the Groupe de recherche en Informatique, Image, automatique et Instrumentation (GREYC) at the University of Caen, 14032 Caen, FRANCE, UMR 6072 CNRS, e-mail: morgan.barbier@unicaen.fr.

Christophe Chabot was with the laboratoire Jean Kuntzmann (LJK), at the University of Joseph Fourier, 38041 Grenoble, FRANCE, UMR 5224 CNRS, e-mail: christophechabotcc@gmail.com.

In the present article we investigate generalized Reed-Solomon codes (denoted by GRS codes in the rest of this paper) over rings with unity. The latter need not be commutative. We show that the main results about GRS codes still hold in this more general situation.

A. Our contributions

In an attempt to build new good codes over $\mathbb{Z}/4\mathbb{Z}$ in a similar way as in [10] we noticed that most results about evaluation codes still hold when we replace the ring of matrices by any noncommutative ring. This generalization is natural to do and permits the study of GRS codes over rings with unity in great generality.

Moreover it allows us to design unique and list decoding algorithms, prove their correctness and study their asymptotic complexities in a very general framework. They are not constraint to have as input a GRS code over a finite field or a Galois ring. They remain valid when the alphabet is any noncommutative ring such as matrices over a finite field or a Galois ring.

In this article we reach the conclusion that GRS codes over finite noncommutative rings are no better than GRS codes over finite commutative rings which are themselves no better than their finite fields counterparts as far as *only* the parameters are concerned.

We summarize our results in the following theorems and propositions which will be proved later in the article.

Theorem 1. *Given three positive integers $k < n \leq q$, let A be a non commutative ring of cardinality q and a GRS code over A of parameters $[n, k, n-k+1]_A$. Then there exists a commutative ring B of cardinality q and a GRS code over B of parameters $[n, k, n-k+1]_B$.*

The same theorem holds when q is a prime power and if we replace "noncommutative rings" by "commutative rings" and "commutative rings" by "finite fields". The "soft-Oh" notation $f(n) \in \tilde{O}(n)$ means that $f(n) \in g(n)\log^{O(1)}(3+g(n))$ (we refer the reader to [23, Chapter 25, Section 7] for details).

Proposition 2. *Given a Galois ring $A = \mathrm{GR}(p^r, s)$ and a RS code over A with parameters $[n, k, n-k+1]_A$, there exists a unique decoding with an asymptotic complexity of $\tilde{O}(rnks\log p)$ bit-operations; and a list decoding algorithm with an asymptotic complexity of $\tilde{O}(n^{r+6}k^5sp^{rs(r-1)})$ bit-operations which can list decode up to the Johnson bound.*

In this paper we provide detailed asymptotic complexities of our decoding algorithms when the alphabet of the RS code is a Galois ring and the ring $\mathbb{F}_q[[t]]/(t^r)$.

Theorem 3. *Given a finite field A, a truncated power series ring B such that $|A| = |B|$, a RS code $\mathcal{C}_A$ over A of parameters $[n, k, n-k+1]_A$ and a unique decoding algorithm* `UDec` *for $\mathcal{C}_A$. Suppose that there exists a RS code $\mathcal{C}_B$ over B of parameters $[n, k, n-k+1]_B$. Then there exists a RS code $\mathcal{C}'_B$ over B of parameters $[n, k, n-k+1]_B$ such that $\mathcal{C}_B/p\mathcal{C}_B = \mathcal{C}'_B/p\mathcal{C}'_B$ and a unique decoding algorithm for $\mathcal{C}'_B$ with a better asymptotic complexity than* `UDec` *as soon as the complexity of* `UDec` *is equal or greater than $\tilde{O}(n)$.*

Note that the asymptotic complexity of the known unique decoding algorithms is at least $\tilde{O}(n)$. In addition, we show that the gain is more significant when the arithmetic of the underlying rings is not done with asymptotically fast algorithms which is the case for practical applications. In this case we have a similar theorem as Theorem 3 for Galois rings.

B. Related work

Our approach for building noncommutative GRS codes is different from the one to build "skew codes" [13]–[16], [19]. Skew polynomial rings over finite fields or Galois rings are used for the construction of codes whose alphabets are finite fields or Galois rings. Here we consider alphabets which are noncommutative rings and not necessarily finite fields. GRS codes over a commutative finite ring have been studied by Marc André Armand in [2], [3]. To our knowledge this paper is the first to study GRS codes over noncommutative rings.

Unique decoding algorithms for RS codes over finite fields have been studied for example in [11], [12], [42]. List decoding algorithms for RS codes over finite fields have been investigated for example in [1], [8], [26], [28], [29], [37], [41]. A unique decoding algorithm for RS codes over Galois rings has been proposed by [4] while list decoding algorithms have been investigated in [2], [3], [5], [6].

A lifting decoding scheme has been first proposed in [24] then in [9], [17]. In this paper we generalize the lifting decoding scheme to obtain unique and list decoding algorithms for GRS code over noncommutative rings.

II. Prerequisites

In this article we let A be a (*not necessarily* commutative) ring with unity, denoted by 1 or 1_A, such that for all $a, b \in A$

- $1.a = a.1 = a$,
- $a.b = 1 \Longrightarrow b.a = 1$.

If $ab = 1$, we say that a is *invertible* or that a is a *unit* whose inverse is b. In this paper, we will only consider rings verifying the above conditions and by "ring" we mean a *not necessarily commutative* ring unless stated otherwise. We denote by $A^\times$ the *not necessarily commutative* group of units of A. An element $a \in A$ is *right-regular* if $ab = 0$ implies $b = 0$ for all $b \in A$. Similarly a is *left-regular* if $ba = 0$ implies $b = 0$ for all $b \in A$. If a is both left- and right-regular we say that a is regular. Note that when A is finite, $A^\times$ coincides with the group of regular elements of A. Suppose now that we have $c = ab$ for three elements of A. Then a is called a *left-divisor* of c and b is called a *right-divisor* of c.

Definition 4 (Commutative subset)**.** A subset S of A is called a *commutative subset* if for each $s, t \in S$ we have $st = ts$.

Definition 5 (Subtractive subset)**.** We borrow the terminology of [33, Definition 2.2 page 3] and say that a subset S of A is *subtractive* if for all $s, t \in S$ with $s \neq t$ we have $s - t \in A^\times$.

Let $A[X]$ be the ring of polynomials over A and, for a *positive* integer k let $A[X]_{<k}$ be the bimodule of all polynomials of $A[X]$ of degree at most $k-1$. Then $A[X]$ is commutative if and only if A is. We denote by $Z(A)$ the center of A. We have $Z(A[X]) = Z(A)[X]$.

Definition 6 (Evaluation map)**.** Let

$$f = \sum_0^l f_i X^i \in A[X]$$

and $a \in A$. We define the *evaluation* of f at a, denoted by $f(a)$, to be the element

$$\sum_0^l f_i a^i \in A.$$

In general, the evaluation map $f \mapsto f(a)$ is not a ring homomorphism. Note however that $f \mapsto f(a)$ is a ring homomorphism whenever $a \in Z(A)$. Let $g \in A[X]$ and suppose that the subset of A constituted by the coefficients of g and a is commutative. Then we have $(fg)(a) = f(a)g(a)$. We call a a *root* of f if $f(a) = 0$.

Let $f \in A[X]$ and $x = (x_1, \ldots, x_n) \in A^n$. For convenience sake the vector $(f(x_1), f(x_2), \ldots, f(x_n))$ of A^n will be denoted by $f(x)$. We include in this section propositions about evaluation and interpolation of polynomials of $A[X]$.

Lemma 7. *Let f and g be nonzero polynomials over A such that the leading coefficient of g is a unit of A. Then there exist unique polynomials $q_l(X), r_l(X) \in A[X]$ such that $f = q_l g + r_l$ and $\deg r_l < \deg g$; and unique polynomials $q_r(X), r_r(X) \in A[X]$ such that $f = g q_r + r_r$ and $\deg r_r < \deg g$.*

Remark 8. Taking the same notations as in Lemma 7 note that we have $q_l = q_r$ and $r_l = r_r$ whenever the coefficients of f and of q_r *or* q_l form a commutative subset of A. It is the case in particular when $g \in Z(A[X])$. A direct consequence of Lemma 7 is that $a \in A$ is a root of f if and only if $X - a$ is a right-divisor of f. Moreover if $a \in Z(A)$ then a is root of f if and only if $X - a$ is a right- and left-divisor of f. The following corollary is the key ingredient of many proofs in this paper.

Corollary 9. *Let f be a polynomial over A of degree at most n with at least $n+1$ roots contained in a commutative subtractive subset of A. Then $f = 0$. It is the case in particular when the considered $(n+1)$ roots are in $Z(A)$.*

The next corollary of Lemma 7 allows to do Lagrange interpolation as soon as the points at which interpolation is done are well chosen.

Corollary 10. *Let $\{x_1, \ldots, x_{n+1}\}$ be a commutative subtractive subset of A and $\{y_1, \ldots, y_{n+1}\}$ be a subset of A. Then*

there exists a unique polynomial $f \in A[X]$ of degree at most n such that $f(x_i) = y_i$, for $i = 1, \ldots, n+1$.

Proof: The proof is included to introduce some notations that will be useful later.

The uniqueness is a direct consequence of Corollary 9. Let

$$L_i(X) = \prod_{j \neq i} (X - x_j).$$

Note that if $h \in A[X]$ and $\lambda, x \in A$ then $(\lambda h)(x) = \lambda(h(x))$. Therefore the polynomial

$$f(X) = \sum_{i=1}^{n+1} y_i L_i(x_i)^{-1} L_i(X), \tag{1}$$

verifies $f(x_i) = y_i$ for $i = 1, \ldots, n+1$. ∎

As in the commutative case, we will need to work in a localization of A. However the operation of localization is slightly more complicated. Let S be a multiplicative subset of A *i.e.* S satisfies $s, t \in S \Rightarrow st \in S$. Following [31, Paragraph 1.6, page 43] one can form the ring of right fractions denoted by AS^{-1} under certain conditions such as the *right Ore condition*.

Definition 11 (Right Ore condition)**.** A subset S of A is said to satisfy the right Ore condition if for all $r \in A$ and $s \in S$ there exists $r' \in A$ and $s' \in S$ such that $rs' = sr'$.

We denote by $\mathrm{ass}(S)$ the set $\{a \in A : as = 0 \text{ for some } s \in S\}$.

Proposition 12. *Let S be a multiplicative subset of A satisfying the right Ore condition such that the image of S in the quotient ring $A/\mathrm{ass}(S)$ consists of regular elements of $A/\mathrm{ass}(S)$. Then the ring of right fractions of A with respect to S exists. We will denote it by AS^{-1}.*

Proof: See [31, Theorem 1.12, page 47]. ∎

A. Error correcting codes

As in the case of linear error correcting codes over a finite field, we define a linear error correcting code as a submodule of A^n for a positive integer n. But, as A is not commutative *a priori* we have to define *left-* and *right*-linear error correcting codes.

Definition 13 (Left and right-linear code)**.** Let n be a positive integer. A *left-* (resp. *right-*) *linear error correcting code* is a left- (resp. right-) submodule C of A such that for each $i \in \{1, \ldots, n\}$ there exists an element of C such that its i'th coordinate is nonzero.

The elements of C are called *codewords* while the elements of A^n are called *words*.

If C is a bi-submodule of A^n then it is called a *linear error correcting code*, or simply an *error correcting code* or a *code* if there is no confusion on the linearity of C.

As in the finite field case, we can define the Hamming distance and weight.

Definition 14 (Hamming weight and distance)**.** Let $u \in A^n$. We call the *Hamming weight* of u the number of nonzero entries of u and denote this number by $w(u)$. Now let $v \in A^n$. The *Hamming distance* between u and v is the nonnegative integer $w(u - v)$ and denoted by $d_H(u, v)$. If there is no confusion the Hamming distance between u and v will be simply called the *distance* between u and v and denoted by $d(u, v)$.

Let C be a *subset* of A^n. The *minimum Hamming distance* of C denoted by d_C is defined as

$$d_C = \min_{u,v \in C \text{ and } u \neq v} d(u, v).$$

If there is no confusion the minimum Hamming distance of C is simply called the *minimum distance* of C. Note that if C is an additive subgroup of A^n we have

$$d_C = \min_{u \in C \setminus \{0\}} w(u).$$

Definition 15 (Support)**.** Let $u = (u_1, \ldots, u_n) \in A^n$. The set

$$\{i \in \{1, \ldots, n\} : u_i \neq 0\}$$

is called the *support* of u and denoted by $\mathrm{Supp}(u)$.

Definition 16 (Inner product)**.** Let $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$ be two elements of A^n. The *inner product* of u and v is defined by

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i \in A.$$

Let C be a bi-submodule of A^n. We can define the *left-dual* submodule ${}^{\perp}C$ of A^n to be the left-submodule of A^n defined by

$${}^{\perp}C = \{u \in A^n : \forall c \in C,\ \langle u, c \rangle = 0\},$$

and similarly the *right-dual* submodule $C^{\perp}$ of A^n to be the right-submodule of A^n defined by

$$C^{\perp} = \{u \in A^n : \forall c \in C,\ \langle c, u \rangle = 0\}.$$

A priori there is no reason that ${}^{\perp}C = C^{\perp}$ except under special hypothesis.

Definition 17 (Parity-check matrix)**.** Let C be a code such that ${}^{\perp}C = C^{\perp}$. Suppose moreover that $C^{\perp}$ has a base $(b_1, \ldots, b_s)$ where b_i is a row matrix for $i = 1, \ldots, s$. Then the matrix

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{pmatrix}$$

is called a *parity-check* matrix of C.

B. Galois rings

We recall briefly basic results about Galois rings that will be useful throughout the article. We fix for this subsection a prime number p and two positive integers r and s.

Proposition 18. *Let $\varphi(X), \psi(X) \in (\mathbb{Z}/p^r\mathbb{Z})[X]$ be monic polynomials of degree s, irreducible modulo p. Then we have a ring isomorphism:*

$$\frac{(\mathbb{Z}/p^r\mathbb{Z})[X]}{(\varphi(X))} = \frac{(\mathbb{Z}/p^r\mathbb{Z})[X]}{(\psi(X))}.$$

Proof: See [35, Statements I and II, page 207]. ∎

Definition 19 (Galois ring)**.** With the same notation as Proposition 18, the ring

$$\frac{(\mathbb{Z}/p^r\mathbb{Z})[X]}{(\varphi(X))}$$

is denoted by $\mathrm{GR}(p^r, s)$ and called a *Galois ring*.

Proposition 20. *The Galois ring $A = \mathrm{GR}(p^r, s)$ is a finite local ring whose maximal ideal is generated by p. Its residue field is $\mathbb{F}_{p^s}$. Moreover all the ideals of A are principal and generated by a power of p.*

Proof: The proposition follows from [35, Paragraph 3.5, page 212] and [39, Theorem 7, page 52]. ∎

In order to use Galois rings as a suitable alphabet for our decoding algorithms we need the following proposition.

Proposition 21. *Let $\mathbb{Z}_p$ be the ring of p-adic integers. We let $\mathbb{Z}_{p^s}$ denote an unramified extension of $\mathbb{Z}_p$ of degree s. Then $\mathrm{GR}(p^r, s)$ and $\mathbb{Z}_{p^s}/(p^r)$ are isomorphic as rings.*

Proof: The proposition follows from [35, Paragraph 3.5, page 212] and [39, Theorem 7, page 52]. ∎

C. Complexity model

In order to analyze the performances of our algorithms, we let $\mathsf{I}(n)$ be the time needed to multiply two integers of bit-size at most n in *binary representation*. It is classical ([18], [21], [38]) that we can take $\mathsf{I}(n) \in O(n \log n 2^{\log^* n})$, where $\log^*$ represents the iterated logarithm of n. If A is a commutative ring, we let $\mathsf{M}_A(n)$ be the cost of multiplying two polynomials of degree at most n with coefficients in A in terms of the number of arithmetic operations in A. It is well known ([23, Theorem 8.23, page 240]) that we can take $\mathsf{M}_A(n) \in \tilde{O}(n)$. Thus the bit-cost of multiplying two elements of $\mathbb{F}_{p^n}$ is $\tilde{O}(n \log p)$ where p is a prime number.

Finally, let us recall that the *expected cost* spent by a randomized algorithm is defined as the average cost for a given input over all the possible executions.

III. Generalized Reed-Solomon codes

In this section, we extend the main propositions about GRS codes over a ring. We study their parameters, duality, key equation, weight distribution and the MacWilliams identity.

From now on and until the end of this article we fix three positive integers $k < n$ and $d = n - k + 1$, a commutative subtractive subset $\{x_1, \ldots, x_n\}$ of A, $x = (x_1, \ldots, x_n)$ and $v = (v_1, \ldots, v_n) \in (Z(A)^\times)^n$.

Definition 22 (Generalized Reed-Solomon code)**.** The left-submodule of A^n generated by the vectors of the form

$$(v_1 f(x_1), \ldots, v_n f(x_n)) \in A^n,$$

with $f \in A[X]_{<k}$ is denoted by

$$\mathrm{GRS}_A(v, x, k) = \mathrm{GRS}_A((v_1, \ldots, v_n), (x_1, \ldots, x_n), k)$$

and is called the *generalized Reed-Solomon* code over A of parameters $[v, x, k]$ or simply $[n, k]$ if there is no confusion on v and x. The integer n is called the *code block length* or simply *length* of $\mathrm{GRS}_A(v, x, k)$. The n-tuple $v = (v_1, \ldots, v_n)$ is called the *weight* of $\mathrm{GRS}_A(v, x, k)$. The n-tuple $x = (x_1, \ldots, x_n)$ is called the *support* of the code. When there is no confusion on the ring A, the weight and the support, we will simply write $\mathrm{GRS}(n, k)$ for $\mathrm{GRS}_A(v, x, k)$. The integer k will be called the *pseudo-dimension* of $\mathrm{GRS}(n, k)$ throughout this paper. When $v = (1_A, \ldots, 1_A)$ we call $\mathrm{GRS}_A(v, x, k)$ a Reed-Solomon code and denote it by $\mathrm{RS}_A(v, x, k)$ or simply $\mathrm{RS}(n, k)$ if there is no confusion on the ring A, the weight and the support.

Note that if $\{x_1, \ldots, x_n\} \subseteq Z(A)$ then the left-linear code $\mathrm{GRS}_A(v, x, k)$ defined in Definition 22 is in fact a code (*i.e.* a bi-submodule of A^n).

Proposition 23. *The left-linear code $\mathrm{GRS}(n, k)$ is free and has a left-basis of cardinality k. In particular when A is commutative the pseudo-dimension k of $\mathrm{GRS}(n, k)$ corresponds to its rank.*

Proof: Let

$$\begin{array}{rccc} \mathrm{ev} : & A[X]_{<k} & \longrightarrow & A^n \\ & f(X) & \longmapsto & (v_1 f(x_1), \ldots, v_n f(x_n)). \end{array}$$

Suppose that $\mathrm{ev}(f) = 0$, then $f(x_i) = 0$ for $i = 1, \ldots, n$. Therefore by Corollary 9 we must have $f = 0$ and ev is injective. But $A[X]_{<k}$ is free and has a basis of cardinality k namely $(1, X, X^2, \ldots, X^{k-1})$ hence the proposition. ∎

Corollary 24. *The code $\mathrm{GRS}(n, k)$ has minimum distance $d = n - k + 1$.*

Proof: Denote by d' the minimum distance of $\mathrm{GRS}(n, k)$. Let $g = (X - x_1)(X - x_2) \cdots (X - x_{k-1})$ be of degree $k-1$. As $\{x_1, \ldots, x_n\}$ is a commutative subset of A, we have $g(x_i) = 0$ for $i = 1, \ldots, k-1$ and $g(x_i)$ is a product of units of A by hypothesis, thus $g(x_i) \neq 0$ for $i = k, \ldots, n$. Therefore $d' \leq n - k + 1$.

Suppose now that there exists a polynomial $f \in A[X]_{<k}$ such that $f(x_i) = 0$ for at least k values of i. By Corollary 9 we must have $f = 0$. Thus $d' \geq n - k + 1$. ∎

Proposition 25. *A generator matrix of $\mathrm{GRS}_A(v, x, k)$ is given by*

$$\begin{pmatrix} v_1 & v_2 & \ldots & v_n \\ v_1 x_1 & v_2 x_2 & \ldots & v_n x_n \\ v_1 x_1^2 & v_2 x_2^2 & \ldots & v_n x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ v_1 x_1^{k-1} & v_2 x_2^{k-1} & \ldots & v_n x_n^{k-1} \end{pmatrix}.$$

Proposition 26. *Let S be a commutative subtractive subset of A. Then there exists a* commutative *ring B such that $|B| = |A|$, and a subtractive subset T of B such that $|T| = |S|$.*

Proof: Let $m = |A|$. We write

$$m = \prod_{i=1}^{r} p_i^{l_i},$$

where $l_1, \ldots, l_r$ are positive integers and $p_1, \ldots, p_r$ are prime numbers. We let $Z = Z(A)[S]$ the commutative subring of A generated by the elements of S over $Z(A)$.

We first prove that p_i divides $|Z|$ for $i = 1, \ldots, r$. Suppose that it is not the case, say p_1 does not divide $|Z|$. Then p_1 divides the order of the quotient additive group A/Z. Then by [30, Lemma 6.1 page 33] there exists $a \in A \setminus Z$ such that $p_1 a \in Z$. But $p_1 1_A = p_1 1_Z$ is invertible in Z and we have $a = (p_1 1_Z)^{-1}(p_1 1_Z)a = (p_1 1_Z)^{-1}(p_1 a) \in Z$.

Now Z is a finite commutative ring and we can write by [7, Theorem 8.7, page 90]

$$Z = \prod_{i=1}^{s} \mathfrak{A}_i$$

in a unique way (up to isomorphism) where $\mathfrak{A}_i$ is a finite local commutative ring for $i = 1, \ldots, s$. We must have $s \geq r$: by [35, Theorem 2, page 199] the cardinality of $\mathfrak{A}_i$ is a prime power, thus if $s < r$ it would contradict the fact that p_i divides $|Z|$ for $i = 1, \ldots, r$.

Denote by $\mathfrak{m}_i$ the maximal ideal of $\mathfrak{A}_i$ for $i = 1, \ldots, r$. Since S is a subtractive subset of Z we must have

$$|S| \leq \min_{i \in \{1,\ldots,r\}} |\mathfrak{A}_i/\mathfrak{m}_i| \,. \tag{2}$$

We can assume that the right-hand side of the above inequality is equal to $|\mathfrak{A}_1/\mathfrak{m}_1|$ without loss of generality. There exists a positive integer l such that $|\mathfrak{A}_1/\mathfrak{m}_1| = p_1^l$ and we have $|S| \leq p_1^l \leq |\mathfrak{A}_i| \leq p_i^{l_i}$ for $j = 2, \ldots, r$ and $j = 1$.

Now let $B = \prod_{i=1}^{s} \mathbb{F}_{p_i^{l_i}}$ be the product ring of finite fields. Then Inequality (2) implies that B contains a subtractive subset T such that $|T| = |S|$. ∎

We have proved Theorem 1.

Theorem 27. *For a GRS code over a finite ring A with parameters $[n, k, n-k+1]_A$, there exists a GRS code over a commutative ring B with $|B| = |A|$ and of parameters $[n, k, n-k+1]_B$.*

Proof: It is a direct consequence of Proposition 26. ∎

Theorem 28. *Suppose that A is finite and that $\{x_1, \ldots, x_n\} \subseteq Z(A)$. Then ${}^{\perp}\mathrm{GRS}_A(v, x, k) = \mathrm{GRS}_A(v, x, k)^{\perp} = \mathrm{GRS}_A(v', x, n-k)$ where $v' = (v'_1, \ldots, v'_n)$ with*

$$v'_i = \left(v_i \prod_{j \neq i} (x_i - x_j) \right)^{-1} \in Z(A).$$

Proof: In short: we let $C = \mathrm{GRS}_A(v, x, k)$, $b_i = (v_1 x_1^i, v_2 x_2^i, \ldots, v_n x_n^i) \in Z(A)^n$ for $i = 0, \ldots, k-1$ and $C' = \mathrm{GRS}_A(v', x, n-k)$.

We first prove that $C' \subseteq C^{\perp}$. Let $f(X) \in A[X]_{<k}$ and $g(X) \in A[X]_{<n-k}$. Then $fg \in A[X]_{<n-1}$. According to Equation (1) we have

$$f(X)g(X) = \sum_{i=1}^{n} f(x_i)g(x_i)L_i(x_i)^{-1}L_i(X).$$

Equating coefficients of degree $n-1$, we get

$$0 = \sum_{i=1}^{n} v_i f(x_i) \left(v_i L_i(x_i)\right)^{-1} g(x_i).$$

Hence $C' \subseteq C^{\perp}$. Doing the same with gf instead of fg, we also get $C' \subseteq {}^{\perp}C$.

We now have to prove that $C' = C^{\perp}$. Let $c' = (c'_1, \ldots, c'_n) \in A^n$. Then $c' \in C^{\perp}$ if and only if

$$\begin{pmatrix} v_1 & v_2 & \ldots & v_n \\ v_1 x_1 & v_2 x_2 & \ldots & v_n x_n \\ v_1 x_1^2 & v_2 x_2^2 & \ldots & v_n x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ v_1 x_1^{k-1} & v_2 x_2^{k-1} & \ldots & v_n x_n^{k-1} \end{pmatrix} \begin{pmatrix} c'_1 \\ c'_2 \\ \vdots \\ c'_n \end{pmatrix} = 0. \tag{3}$$

The matrix has its coefficients in the commutative ring $Z(A)$ thus we can compute the determinant of

$$V = \begin{pmatrix} v_1 & v_2 & \ldots & v_k \\ v_1 x_1 & v_2 x_2 & \ldots & v_k x_k \\ v_1 x_1^2 & v_2 x_2^2 & \ldots & v_k x_k^2 \\ \ldots & \ldots & \ldots & \ldots \\ v_1 x_1^{k-1} & v_2 x_2^{k-1} & \ldots & v_k x_k^{k-1} \end{pmatrix}.$$

And we have

$$\det V = \left(\prod_{i=1}^{k} v_i \right) \left(\prod_{i \neq j} (x_i - x_j) \right).$$

This determinant is a unit of $Z(A)$ by the hypothesis made on the weight v and the support x of the code C. Thus V has an inverse in $M_k(Z(A))$ and therefore V^{-1} is also the left- and right-inverse of V in $M_k(A)$. As a consequence given $(c'_{k+1}, \ldots, c'_n) \in A^{n-k}$ there exists one and only one k-tuple $(c'_1, \ldots, c'_k) \in A^k$ such that Equation (3) is satisfied. Thus the number of solutions in A^n of Equation (3) is equal to $|A|^{n-k}$ which is also the number of elements of C'.

Using the fact that $(b_i)_{i=0,\ldots,k-1}$ is a basis of C as a right-module and the following system of equations

$$\left(c'_1, c'_2, \ldots, c'_n\right) \begin{pmatrix} v_1 & v_1 x_1 & \ldots & v_1 x_1^{k-1} \\ v_2 & v_2 x_2 & \ldots & v_2 x_2^{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ v_n & v_n x_n & \ldots & v_n x_n^{k-1} \end{pmatrix} = 0,$$

(which is the transposed system of system (3)) instead of the system (3) of equations, we get $C' = {}^{\perp}C$. ∎

Corollary 29. *Suppose that A is finite and that $\{x_1, \ldots, x_n\} \subseteq Z(A)$. Then there exists a parity-check matrix for $\mathrm{GRS}_A(v, x, k)$ equal to*

$$\begin{pmatrix} v'_1 & v'_2 & \ldots & v'_n \\ v'_1 x_1 & v'_2 x_2 & \ldots & v'_n x_n \\ v'_1 x_1^2 & v'_2 x_2^2 & \ldots & v'_n x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ v'_1 x_1^{n-k-1} & v'_2 x_2^{n-k-1} & \ldots & v'_n x_n^{n-k-1} \end{pmatrix}$$

where the v'_i are defined as in Theorem 28.

Lemma 30 (Goppa formulation for GRS codes)**.** *When A is finite and $\{x_1, \ldots, x_n\} \subseteq Z(A)$, we have, for $c = (c_1, \ldots, c_n) \in A^n$,*

$$c \in \mathrm{GRS}_A(v, x, k) \Longleftrightarrow \sum_{i=1}^{n} \frac{c_i v'_i}{1 - x_i X} = 0 \mod X^{n-k} \tag{4}$$

where the v'_i are defined as in Theorem 28.

Proof: We check that equation 4 makes sense. First, The ideal generated by X^{n-k} is a two sided ideal. Secondly, in $A[[X]]$ the power series $(1-f(X))$ where $f(X) \in XA[[X]]$ has a two-sided inverse given by $\sum_{i=0}^{\infty} f(X)^i X^i$. This can be shown by direct computation.

The rest of the proof is also a direct computation and is left to the reader. ∎

Proposition 31 (Key equation for GRS codes)**.** *Suppose that A is finite and that $\{x_1, \ldots, x_n\} \subseteq Z(A)$. Let $y = (y_1, \ldots, y_n) \in A^n$ be the received word such that there exists a unique codeword $c = (c_1, \ldots, c_n) \in \mathrm{GRS}(n,k)$ at distance at most $\left\lfloor \frac{n-k}{2} \right\rfloor$. We let $e = (e_1, \ldots, e_n) = y - c$ and $E = \mathrm{Supp}(e)$. Let*

$$\sigma(X) = \prod_{i \in E} (1 - x_i X),$$

$$\omega(X) = \sum_{i \in E} e_i v_i' \left(\prod_{j \in E \text{ and } j \neq i} (1 - x_j X) \right)$$

and

$$S(X) = \sum_{i=1}^{n} \frac{y_i v_i'}{1 - x_i X} \mod X^{n-k}.$$

Then we have $\sigma(X)S(X) = \omega(X) \mod X^{n-k}$.

Proof: This is a direct computation using Lemma 30. ∎

The following proposition will be useful to compute the complexities of the proposed algorithms and to prove MacWilliams identity for GRS codes over non commutative rings.

Proposition 32. *Suppose that A is finite. Let $\mathcal{C}_s$ be the number of codewords from $\mathrm{GRS}(n,k)$ of weight s. Then $\mathcal{C}_0 = 1$, $\mathcal{C}_s = 0$ for $1 \leq s \leq d-1$ and*

$$\mathcal{C}_s = \binom{n}{s} \sum_{i=0}^{s-n+k} (-1)^i \binom{s}{i} \left(|A|^{s-n+k+1-i} - 1 \right) \quad (5)$$

for $d \leq s \leq n$.

Proof: The result is obvious for $\mathcal{C}_0$ and $\mathcal{C}_s$ for $1 \leq s \leq d-1$. So now let $d \leq s \leq n$ and denote by $N(i_1, \ldots, i_s)$ the number of codewords with zeros at coordinates $i_1, \ldots, i_s$. Then $|N(i_1, \ldots, i_s)| = |A|^{k-s}$ by Remark 8 and Corollary 9. The rest of the proof is identical to [34, paragraph 3.9, page 79]. ∎

Proposition 33. *Suppose that A is finite and that $\{x_1, \ldots, x_n\} \subseteq Z(A)$. With the notations of Proposition 32, let $\mathcal{C} = \mathrm{GRS}_A(v, x, k)$,*

$$W_{\mathcal{C}}(X,Y) = \sum_{i=0}^{n} \mathcal{C}_i X^{n-i} Y^i,$$

and

$$W_{\mathcal{C}^\perp}(X,Y) = \sum_{i=0}^{n} (\mathcal{C}^\perp)_i X^{n-i} Y^i.$$

Then

$$W_{\mathcal{C}}(X,Y) = \frac{1}{|A|^{n-k}} W_{\mathcal{C}^\perp}(X + (|A|-1)Y, X - Y).$$

Proof: By Proposition 26 there exists a RS code over a commutative ring B of cardinality $|A|$. We denote by $\mathcal{D}$ this code. It has parameters $[n, k, n-k+1]_B$ over B. We can take B to be a product of finite fields by 26 again. Seen as module over itself, B is semisimple and thus by [40, Paragraph 1, page 989] is injective. Now by [44, Theorem 1.2 page 4] and [44, Remark 1.3, page 4] B is a Frobenius ring and [44, Theorem 8.3 page 18] can be applied. By Proposition 32 the weight distribution of $\mathcal{C}$ is the same as the one of $\mathcal{D}$.

Gathering the above results we finally get

$$\begin{aligned} W_{\mathcal{C}}(X,Y) &= W_{\mathcal{D}}(X,Y) \\ &= \frac{1}{|B|^{n-k}} W_{\mathcal{D}^\perp}(X + (|B|-1)Y, X - Y) \\ &= \frac{1}{|A|^{n-k}} W_{\mathcal{C}^\perp}(X + (|A|-1)Y, X - Y). \end{aligned}$$

∎

IV. Unique decoding of generalized Reed-Solomon codes

A. Unique decoding over certain valuation rings

In this section we design a unique decoding algorithm for GRS codes over a discrete valuation ring. We suppose that A has the following property:

- $(*)$ *there exists a regular element p which is not a unit such that $p \in Z(A)$ and such that every element $a \in A$ can be uniquely written as $\sum_{i=0}^{\infty} a_i p^i$ where, for all $i \in \mathbb{N}$, a_i is in a set of representatives of $A/(p)$.*

It is the case for example for the two rings $\mathbb{Z}_q$ and $M_\ell(\mathbb{Z}_q)$ where $\mathbb{Z}_q$ denotes an unramified extension of the p-adic numbers or for the power series ring $\kappa[[t]]$ and the ring of matrices $M_\ell(\kappa[[t]])$. We will need in this section and the rest of this paper to divide elements of A by p, which is provided by the following lemma.

Lemma 34. *The ring of right fractions with respect to the subset S of A formed by the powers of p exists.*

Proof: Clearly S is a multiplicative subset of A. The fact that p is in the center of A and regular implies that S satisfies the right Ore condition (Definition 11) and that $\mathrm{ass}(S) = \{0\}$. Therefore we can apply Proposition 12. ∎

Lemma 34 in combination with [31, Paragraph 1.3 (iii), page 42] shows that we have a natural injection $A \subseteq AS^{-1}$ and we will freely use this injection to identify elements of A and their images in the ring AS^{-1}.

We let $\Lambda = A/(p)$ be the residual ring of A. In the sequel for a vector $b = (b_1, \ldots, b_n) \in A^n$, we will denote by $b \mod p$ the vector $(b_1 \mod p, \ldots, b_n \mod p) \in (A/(p))^n$ and if $f \in A[X]$, recall that we denote by $f(b)$ the vector $(f(b_1), \ldots, f(b_n)) \in A^n$.

Let $\mathcal{C} = \mathrm{GRS}_A(v, x, k)$ be a generalized Reed-Solomon code. Then $\mathcal{C}/p^r\mathcal{C} = \mathrm{GRS}_{A/(p^r)}(v \mod p^r, x \mod p^r, k)$.

Lemma 35. *Let $c \in \mathcal{C}$ such that $c/p \in A^n$. Then $c/p \in \mathcal{C}$.*

Proof: Let $f \in A[X]_{<k}$ such that $c = f(x)$. Then $f(x) \mod p = 0$. By Corollary 9 we have $f \mod p = 0$ and there exists $g \in A[X]_{<k}$ such that $f(X) = pg(X)$. ∎

We now give an algorithm of unique decoding for GRS codes over $B = A/(p^r)$ for a positive integer r. We let $\tau = \left\lfloor \frac{n-k}{2} \right\rfloor$. The idea of the algorithm is to do a Hensel lifting. We first look at the received word modulo p. Then we call a decoding algorithm for the GRS code modulo p. It then returns the component of degree 0 of the wanted codeword and the error. We subtract these to the received word and then can divide the result by p to reiterate the process and obtain the component of degree 1, 2 up to $r-1$. We first precise what is the black box algorithm.

Algorithm 1 Black box unique decoding algorithm

Input: a received vector y of Λ^n with at most $\left\lfloor \frac{n-k}{2} \right\rfloor$ errors.
Output: the message $m \in \Lambda^k$ such that the corresponding codeword is within distance $\left\lfloor \frac{n-k}{2} \right\rfloor$ of y and the error $e \in \Lambda^n$.

In the following algorithm we denote by G a generator matrix of $\mathcal{C}$.

Algorithm 2 Unique decoding over a valuation ring

Input: a received vector y of B^n with at most τ errors, and a **black box** unique decoding algorithm for $\mathcal{C}/p\mathcal{C}$ as Algorithm 1.
Output: the unique codeword of $\mathcal{C}/p^r\mathcal{C}$ within distance τ of y.

1: Compute $y_0 \in A$ such that $y_0 \mod p^r = y$.
2: **for** $i = 0 \to r-1$ **do**
3: Call the **black box** with input $y_i \mod p$ and obtain $m_i \in \Lambda^k$ and $e'_i \in \Lambda^n$.
4: $c_i \leftarrow m_i G \in \mathcal{C}$.
5: $e_i \leftarrow$ a representative of e'_i such that $\mathrm{Supp}(e_i) = \mathrm{Supp}(e')$.
6: $y_{i+1} \leftarrow (y_i - c_i - e_i)/p$.
7: **end for**
8: **return** $\sum_{i=0}^{r-1} p^i c_i \mod p^r$.

Proposition 36. *Algorithm 2 is correct and can decode up to τ errors.*

Proof: We let $c \in \mathcal{C}$ and $e \in A^n$ be such that $w(e) \leq \tau$ and $y = c + e$. We will show by induction on i that the following items holds after step 7 of Algorithm 2:

1) $y_0 = \sum_{j=0}^{i} p^j(c_j + e_j) + p^{i+1} y_{i+1}$,
2) $y_{i+1} = (y_i - c_i - e_i)/p = c'' + e''$ where $c'' \in \mathcal{C}$ and $e'' \in A^n$ with $\mathrm{Supp}(e'') \subseteq \mathrm{Supp}(e)$.

For $i = 0$ item 1 is trivially satisfied and we have at step 4

$$y \mod p = (c+e) \mod p = c' + e'$$

where $c' \in \mathcal{C}/p\mathcal{C}$ and $e' \in \Lambda^n$ such that $w(e') \leq \tau$. By unicity of c' we must have $c \mod p = c'$ and $e \mod p = e'$. Therefore we have after step 6

$$(y_0 - c_0 - e_0)/p = (c - c_0)/p + (e - e_0)/p$$

and by Lemma 35 we have $(c - c_0)/p \in \mathcal{C}$. Moreover $\mathrm{Supp}(e_0) \subseteq \mathrm{Supp}(e)$ thus item 2 is satisfied.

Now suppose that the induction holds for $i \geq 0$. Then we get from item 2 of the inductive hypothesis and from step 4

$$y_{i+1} \mod p = (c'' + e'') \mod p = c' + e'$$

where $c'' \in \mathcal{C}$ and $e'' \in A^n$ such that $\mathrm{Supp}(e'') \subseteq \mathrm{Supp}(e)$, thus $w(e'') \leq \tau$. Therefore by unicity of c' we must have $c'' \mod p = c'$ and $e'' \mod p = e'$. Therefore we deduce

$$(y_{i+1} - c_{i+1} - e_{i+1})/p = (c'' - c_{i+1})/p + (e'' - e_{i+1})/p$$

and by Lemma 35 $(c'' - c_{i+1})/p \in \mathcal{C}$. Moreover $\mathrm{Supp}(e_{i+1}) \subseteq \mathrm{Supp}(e'') \subseteq \mathrm{Supp}(e)$ and item 2 is satisfied. As for item 1, we have

$$\begin{aligned} p^{i+2} y_{i+2} &= p^{i+1}(y_{i+1} - c_{i+1} - e_{i+1}) \\ &= y_0 - \sum_{j=0}^{i} p^j(c_j + e_j) - p^{i+1}(c_{i+1} + e_{i+1}). \end{aligned}$$

Now taking $i = r$, we get

$$y = y_0 \mod p^r = \sum_{j=0}^{r-1} p^j c_j + \sum_{j=0}^{r-1} p^j e_j \mod p^r.$$

We have $c_j \in \mathcal{C}$ for $j = 1, \ldots, r-1$ thus

$$\sum_{j=0}^{r-1} p^j c_j \mod p^r \in \mathcal{C}/p^r\mathcal{C}.$$

As $\mathrm{Supp}(e_j) \subseteq \mathrm{Supp}(e)$ for $j = 1, \ldots, r-1$ we have

$$w\left(\sum_{j=0}^{r-1} p^j e_j \mod p^r \right) \leq \tau.$$

Therefore the unicity of c implies that

$$c = \sum_{j=0}^{r-1} p^j c_j \mod p^r.$$

■

Proposition 37. *Let $\mathsf{UDec}(\mathcal{C})$ be the complexity of the black box decoding algorithm for $\mathcal{C}/p\mathcal{C}$ in terms of the number of bit-operations given as input of Algorithm 2 and $\mathsf{Lift}(\mathcal{C})$ the complexity of lifting a codeword from $\mathcal{C}/p\mathcal{C}$ into $\mathcal{C}$ (*i.e. *the bit-cost of step 4 of Algorithm 2). Then Algorithm 2 performs a number of $r(\mathsf{UDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C}))$ bit-operations.*

Lemma 38. *Suppose that $v = (1_B, \ldots, 1_B)$. If $B = \mathrm{GR}(p^r, s)$ we can take $\mathsf{Lift}(\mathcal{C}) = O(nk\mathsf{M}_{\mathbb{F}_p}(s)\mathsf{I}(\log p))$ bit-operations. If $B = \mathbb{F}_{p^s}[[t]]/(t^r)$ we can take $\mathsf{Lift}(\mathcal{C}) = O(nk)$ arithmetic operations in $\mathbb{F}_{p^s}$; in the situation where the support of $\mathcal{C}$ is contained in $\mathbb{F}_{p^s}$ we can take $\mathsf{Lift}(\mathcal{C}) = O(M_{\mathbb{F}_{p^s}}(n) \log n)$ arithmetic operations in $\mathbb{F}_{p^s}$.*

Proof: Lifting a codeword from $\mathcal{C}/p\mathcal{C}$ into $\mathcal{C}$ can be done by the matrix-vector product mG where G is a generator matrix of $\mathcal{C}$ and

- $m \in B^k$ is a representative of the message modulo p whose coefficients have the same bit-size as elements of $\mathbb{F}_p$ when $B = \mathrm{GR}(p^r, s)$,
- $m \in \mathbb{F}_{p^s}$ is a representative of the message modulo t when $B = \mathbb{F}_{p^s}[[t]]/(t^r)$.

In the situation where $B = \mathbb{F}_{p^s}[[t]]/(t^r)$ and that the support of $\mathcal{C}$ is included in $\mathbb{F}_{p^s}$ we can use fast multipoint evaluation of a polynomial of degree at most n with coefficients in $\mathbb{F}_{p^s}$ in n points of $\mathbb{F}_{p^s}$ which is done in $O(M_{F_{p^s}}(n)\log n)$ by [23, Corollary 10.8, page 295]. ∎

Corollary 39. *Suppose that B is the ring $\mathbb{F}_{p^s}[[t]]/(t^r)$. Then there exists a unique decoding algorithm for $GRS_A(v,x,k)$ where $x \in \mathbb{F}_{p^s}$ with an asymptotic complexity of $\tilde{O}(rn)$ arithmetic operations in $\mathbb{F}_{p^s}$.*

Proof: This is a direct consequence of Proposition 37, Lemma 38 and [27]. ∎

Remark 40. Taking the notations as Corollary 39 note that, given any GRS code C over B, we can always find a GRS code C' over B with same parameters as C such that its support is included in $\mathbb{F}_{p^s}$.

Theorem 41. *Given a finite field A, a truncated power series ring B such that $|A| = |B|$, a RS code $\mathcal{C}_A$ over A of parameters $[n,k,n-k+1]_A$ and a unique decoding algorithm* `UDec` *for $\mathcal{C}_A$. Suppose that there exists a RS code $\mathcal{C}_B$ over B of parameters $[n,k,n-k+1]_B$. Then there exists a RS code $\mathcal{C}'_B$ over B of parameters $[n,k,n-k+1]_B$ such that $\mathcal{C}_B/p\mathcal{C}_B = \mathcal{C}'_B/p\mathcal{C}'_B$ and a unique decoding algorithm for $\mathcal{C}'_B$ with a better asymptotic complexity than* `UDec` *as soon as the complexity of* `UDec` *is equal or greater than* $\mathsf{Lift}(\mathcal{C}'_B)$.

Note that the classical unique decoding algorithms over a finite fields $\mathbb{F}$ have a complexity of $O(\mathsf{M}_{\mathbb{F}}(n)\log n)$ so that the theorem holds when `UDec` is the algorithm of [22] or of [27].

Corollary 42. *Suppose that B is the Galois ring $\mathrm{GR}(p^r,s)$. Then there exists a unique decoding algorithm for $GRS_A(v,x,k)$ with an asymptotic complexity of $\tilde{O}(rnks\log p)$ bit-operations.*

Proof: This is a direct consequence of Proposition 37, Lemma 38 and [27]. ∎

Remark 43. We show that the gain is more significant when the arithmetic of the underlying rings and fields is not done with asymptotically fast algorithms which is the case for practical applications. By [20, Table 1, page 5] the complexity of the unique decoding black box algorithm is $O(n^2)$ arithmetic operations over the alphabet when the latter is a finite field of characteristic 2.

- If $B = \mathrm{GR}(p^r,s)$ then Algorithm 2 performs at most $O(rn^2s^2\log^2 p)$ bit-operations.
- If $B = \mathbb{F}_{p^s}[[t]]/(t^r)$ then Algorithm 2 performs at most $O(rn^2s^2)$ arithmetic operations over $\mathbb{F}_p$.

Note that if B is a finite field of cardinality p^{rs} the cost of unique decoding is $O(n^2r^2s^2\log^2 p)$ bit-operations. The unique decoding becomes cheaper when the alphabet is a ring and we have a similar theorem as Theorem 41 for Galois rings.

Example 44. Let now $\mathcal{C}$ be the RS code over $\mathbb{Z}/11^3\mathbb{Z}$ with support $(1,2,3,4,5,6,7)$ of dimension 3. Thus $\mathcal{C}$ has parameters $[7,3,5]_{\mathbb{Z}/11^3\mathbb{Z}}$ by Corollary 24. Therefore its unique decoding radius is 2. Let $y = (133,158,163,181,201,344,247)$ be a received word. Executing algorithm 2 we get the following:

1) $i = 0$ and $y_0 = y \mod 11 = (1,4,9,5,3,3,5) \in \mathbb{F}_{11}$.
   The black box algorithm returns
   - $c_0 \mod p = (1,4,9,5,3,3,5) \in \mathcal{C}/11\mathcal{C} \subseteq \mathbb{F}_{11}^7$ and
   - $e_0 \mod p = (0,0,0,0,0,0,0) \in \mathbb{F}_{11}^7$,

   which can be lifted to
   - $c_0 = (1,4,9,16,25,36,49) \in \mathcal{C} \subseteq (\mathbb{Z}/11^3\mathbb{Z})^7$ and
   - $e_0 = (0,0,0,0,0,0,0) \in (\mathbb{Z}/11^3\mathbb{Z})^7$.
2) $i = 1$ and $y_1 = (y_0 - c_0 - e_0)/11 \mod 11 = (1,3,3,4,5,6,7)$.
   The black box algorithm returns
   - $c_1 \mod p = (1,2,3,4,5,6,7) \in \mathcal{C}/11\mathcal{C}$ and
   - $e_1 \mod p = (0,1,0,0,0,0,0) \in \mathbb{F}_{11}^7$,

   which can be lifted to
   - $c_1 = (1,2,2,4,5,6,7) \in \mathcal{C}$ and
   - $e_1 = (0,1,0,0,0,0,0) \in (\mathbb{Z}/11^3\mathbb{Z})^7$.
3) $i = 2$ and $y_2 = (y_1 - c_1 - e_1)/11 \mod 11 = (1,1,1,1,1,2,1)$.
   The black box algorithm returns
   - $c_2 \mod p = (1,1,1,1,1,1,1) \in \mathcal{C}/11\mathcal{C}$ and
   - $e_2 \mod p = (0,0,0,0,0,1,0) \in \mathbb{F}_{11}^7$,

   which can be lifted to
   - $c_2 = (1,1,1,1,1,1,1) \in \mathcal{C}$ and
   - $e_2 = (0,1,0,0,0,1,0) \in (\mathbb{Z}/11^3\mathbb{Z})^7$.
4) We then build the codeword from its homogeneous components $y_0 + 11y_1 + 11^2y_2 \mod 11^3 = (133,147,163,181,201,223,247)$.

In this example the error is $e = e_0 + 11e_1 + 11^2e_2 \mod 11^3 = (0,11,0,0,0,11^2,0)$.

B. The Welch-Berlekamp algorithm

Before giving the Welch-Berlekamp decoding algorithm, we need to define what the *evaluation* of a bivariate polynomial over A is. Let $Q = \sum Q_{i,j}X^iY^j \in A[X,Y]$ be such a polynomial. We define the evaluation of Q at $(a,b) \in A^2$ to be

$$Q(a,b) = \sum Q_{i,j}b^ja^i \in A.$$

Be careful of the order of a and b. This choice will be explained in the proof of Lemma 45. Let $f \in A[X]$, we define the evaluation of Q at f to be

$$Q(X,f(X)) = \sum Q_{i,j}(f(X))^jX^i \in A[X].$$

As in the univariate case, the evaluation maps defined above are not ring homomorphisms in general.

Lemma 45. *Let $g \in A[X]$, $Q \in A[X,Y]$ of degree at most 1 in Y and $z \in A$. Then*

$$(Q(X,g(X)))(z) = Q(z,g(z)).$$

Proof: We write

$$\begin{aligned} Q(X,Y) &= Q_0(X) + Q_1(X)Y \\ &= Q_0(X) + \left(\sum_i Q_{1i}X^i\right)Y. \end{aligned}$$

The proof is an easy calculation:

$$\begin{aligned}(Q(X,g(X)))(z) &= \left(Q_0(X)+\sum_i Q_{1i}g(X)X^i\right)(z)\\ &= Q_0(z)+\sum_i Q_{1i}g(z)z^i\\ &= Q(z,g(z)) \text{ by definition.}\end{aligned}$$

■

We now adapt the Welch-Berlekamp algorithm [12] to noncommutative GRS. By Corollary 24, we have $\left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor$. We let $\tau = \left\lfloor \frac{n-k}{2} \right\rfloor$.

Algorithm 3 Welch-Berlekamp

Input: a received vector y of A^n with at most τ errors.
Output: the unique codeword within distance τ of y.

1: $y' \leftarrow (v_1^{-1}y_1, \ldots, v_n^{-1}y_n)$.
2: Find $Q = Q_0(X) + Q_1(X)Y \in (A[X])[Y]$ of degree 1 such that
   1) $Q(x_i, y_i') = 0$ for all $1 \le i \le n$,
   2) $\deg Q_0 \le n - \tau - 1$,
   3) $\deg Q_1 \le n - \tau - 1 - (k-1)$.
   4) The leading coefficient of Q_1 is 1_A.
3: **return** The unique root of Q in $A[X]_{<k}$.

In order to prove the correctness of the Welch-Berlekamp algorithm, we start with the following lemmas.

Lemma 46. *Let $y = (y_1, \ldots, y_n) \in A^n$ be such that $\tau \le \left\lfloor \frac{n-k}{2} \right\rfloor$. Then there exists a nonzero bivariate polynomial $Q = Q_0 + Q_1 Y \in A[X,Y]$ satisfying*

1) $Q(x_i, y_i) = 0$ *for* $i = 1, \ldots, n$.
2) $\deg Q_0 \le n - \tau - 1$.
3) $\deg Q_1 \le n - \tau - 1 - (k-1)$.
4) *The leading coefficient of Q_1 is 1_A.*

Proof: We can write $y = c + e$ uniquely with $c = f(x) \in \mathrm{GRS}(n,k)$ for a polynomial $f \in A[X]_{<k}$ and $e \in A^n$, $w(e) \le \tau$. Let $E = \mathrm{Supp}(e)$ and $Q_1(X) = \prod_{i \in E}(X - x_i)$. Then take $Q_0(X) = -f(X)Q_1(X)$. Then $Q(X,Y) = Q_0(X) + Q_1(X)Y$ satisfies the conditions of the lemma. ■

Lemma 47. *Let $Q \in A[X,Y]$ be a bivariate polynomial satisfying the four conditions of Lemma 46 and $f \in A[X]_{<k}$ be such that $d(y, f(x)) \le \tau$. Then $Q(X, f(X)) = 0$.*

Proof: The polynomial $Q(X, f(X))$ has degree at most $n - \tau - 1$. By Lemma 45 we have $(Q(X, f(X)))(x_i) = Q(x_i, f(x_i)) = Q(x_i, y_i) = 0$ for at least $n - \tau$ values of $i \in \{1, \ldots, n\}$. And by Corollary 9 we must have $Q(X, f(X)) = 0$. ■

The correctness of the algorithm is a direct consequence of Lemma 46 and 47.

Proposition 48. *Algorithm 3 works correctly as expected and can correct up to $\left\lfloor \frac{n-k}{2} \right\rfloor$ errors.*

Example 49. Let $\mathcal{C}$ be the RS code over $M_2(\mathbb{F}_7)$ with support

$$\left(\begin{pmatrix}1&0\\0&1\end{pmatrix};\begin{pmatrix}2&0\\0&2\end{pmatrix};\begin{pmatrix}3&0\\0&3\end{pmatrix};\begin{pmatrix}4&0\\0&4\end{pmatrix};\begin{pmatrix}5&0\\0&5\end{pmatrix}\right)$$

of dimension 3. Thus $\mathcal{C}$ is a $[5,3,2]_{M_2(\mathbb{F}_7)}$ linear code by Corollary 24. Therefore its unique decoding radius is 2. Let

$$y = \left(\begin{pmatrix}5&3\\2&2\end{pmatrix};\begin{pmatrix}2&2\\4&1\end{pmatrix};\begin{pmatrix}4&0\\0&4\end{pmatrix};\begin{pmatrix}2&3\\3&3\end{pmatrix};\begin{pmatrix}5&5\\0&6\end{pmatrix}\right)$$

be a received word. Executing algorithm 3 we get the following:

1) By Lemma 46, Q is found using linear algebra with the affine systems of equations

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&5&3&5&3\\
0&1&0&1&0&1&0&1&2&2&2&2\\
1&0&2&0&4&0&1&0&2&2&4&4\\
0&1&0&2&0&4&0&1&4&1&1&2\\
1&0&3&0&2&0&6&0&4&0&5&0\\
0&1&0&3&0&2&0&6&0&4&0&5\\
1&0&4&0&2&0&1&0&2&3&1&5\\
0&1&0&4&0&2&0&1&3&3&5&5\\
1&0&5&0&4&0&6&0&5&5&4&4\\
0&1&0&5&0&4&0&6&0&6&0&2
\end{pmatrix}
\times
\begin{pmatrix}
a_0&b_0\\u_0&v_0\\a_1&b_1\\u_1&v_1\\a_2&b_2\\u_2&v_2\\a_3&b_0\\u_3&v_3\\a_4&b_4\\u_4&v_4\\1&0\\0&1
\end{pmatrix} = 0.$$

From this we find a solution and therefore polynomials

$$\begin{aligned}Q_0(X) &= \begin{pmatrix}a_0&b_0\\u_0&v_0\end{pmatrix} + \begin{pmatrix}a_1&b_1\\u_1&v_1\end{pmatrix}X + \begin{pmatrix}a_2&b_2\\u_2&v_2\end{pmatrix}X^2\\ &\quad + \begin{pmatrix}a_3&b_3\\u_3&v_3\end{pmatrix}X^3\\ &= \begin{pmatrix}2&1\\2&6\end{pmatrix} + \begin{pmatrix}0&5\\0&3\end{pmatrix}X + \begin{pmatrix}2&4\\0&5\end{pmatrix}X^2 +\\ &\quad + \begin{pmatrix}6&3\\2&4\end{pmatrix}X^3.\end{aligned}$$

and

$$\begin{aligned}Q_1(X) &= \begin{pmatrix}a_4&b_4\\u_4&v_4\end{pmatrix} + \begin{pmatrix}1&0\\0&1\end{pmatrix}X\\ &= \begin{pmatrix}4&0\\0&4\end{pmatrix} + \begin{pmatrix}1&0\\0&1\end{pmatrix}X.\end{aligned}$$

2) $Q(X,Y) = Q_0(X) + Q_1(X)Y$ has only one root in $M_2(\mathbb{F}_7)[X]$ by Lemma 7. It is computed with the classical Euclidean division algorithm. Thus we get the following root of Q

$$\begin{pmatrix}3&5\\3&2\end{pmatrix} + \begin{pmatrix}1&1\\1&4\end{pmatrix}X + \begin{pmatrix}1&4\\5&3\end{pmatrix}X^2.$$

And then retrieve the corresponding codeword

$$c = \left(\begin{pmatrix} 5 & 3 \\ 2 & 2 \end{pmatrix}; \begin{pmatrix} 2 & 2 \\ 4 & 1 \end{pmatrix}; \begin{pmatrix} 1 & 2 \\ 2 & 6 \end{pmatrix}; \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}; \begin{pmatrix} 5 & 5 \\ 0 & 6 \end{pmatrix} \right).$$

We can modify Algorithm 3 so that it also returns the error. In this example the error is

$$e = \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 3 & 5 \\ 5 & 5 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

We now give an example of a Reed-Solomon code defined over $M_2(\mathbb{Z}/5^2\mathbb{Z})$. We use Algorithm 2 together with Algorithm 3 to decode a word.

Example 50. Let $A = M_2(\mathbb{Z}/5^2\mathbb{Z})$. The ideal generated by $p = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$ is two sided and p satisfies the condition $(*)$ of Subsection IV-A. Therefore we can apply Algorithm 2 to the Reed-Solomon code whose support is

$$\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}; \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}; \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \right)$$

and of dimension 2. Let

$$y = \left(\begin{pmatrix} 21 & 14 \\ 14 & 22 \end{pmatrix}; \begin{pmatrix} 20 & 8 \\ 15 & 7 \end{pmatrix}; \begin{pmatrix} 5 & 17 \\ 5 & 1 \end{pmatrix}; \begin{pmatrix} 22 & 6 \\ 13 & 3 \end{pmatrix} \right)$$

be a received word. Executing Algorithm 2 we get

1) $i = 0$ and

$$y_0 \mod p = y \mod p = \left(\begin{pmatrix} 1 & 4 \\ 4 & 2 \end{pmatrix}; \begin{pmatrix} 0 & 3 \\ 0 & 2 \end{pmatrix}; \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix}; \begin{pmatrix} 2 & 1 \\ 3 & 3 \end{pmatrix} \right).$$

Algorithm 3 with input y_0 returns

$$c_0 \mod p = \left(\begin{pmatrix} 1 & 4 \\ 4 & 2 \end{pmatrix}; \begin{pmatrix} 3 & 3 \\ 2 & 4 \end{pmatrix}; \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix}; \begin{pmatrix} 2 & 1 \\ 3 & 3 \end{pmatrix} \right)$$

and the error

$$e_0 \mod p = \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 2 & 0 \\ 3 & 3 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right)$$

which can be lifted to the codeword

$$c_0 = \left(\begin{pmatrix} 6 & 4 \\ 4 & 2 \end{pmatrix}; \begin{pmatrix} 8 & 8 \\ 7 & 4 \end{pmatrix}; \begin{pmatrix} 10 & 12 \\ 10 & 6 \end{pmatrix}; \begin{pmatrix} 12 & 16 \\ 13 & 8 \end{pmatrix} \right).$$

We then compute

$$y_1 = (y_0 - c_0 - e_0)/p = \left(\begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}; \begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix}; \begin{pmatrix} 4 & 1 \\ 4 & 4 \end{pmatrix}; \begin{pmatrix} 2 & 3 \\ 0 & 4 \end{pmatrix} \right).$$

2) $i = 1$ and

$$y_1 \mod p = \left(\begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}; \begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix}; \begin{pmatrix} 4 & 1 \\ 4 & 4 \end{pmatrix}; \begin{pmatrix} 2 & 3 \\ 0 & 4 \end{pmatrix} \right).$$

Algorithm 3 with input $y_1 \mod p$ returns

$$c_0 \mod p = \left(\begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}; \begin{pmatrix} 1 & 4 \\ 3 & 4 \end{pmatrix}; \begin{pmatrix} 4 & 1 \\ 4 & 4 \end{pmatrix}; \begin{pmatrix} 2 & 3 \\ 0 & 4 \end{pmatrix} \right)$$

and the error

$$e_0 \mod p = \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right)$$

which can be lifted to the codeword

$$c_0 = \left(\begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}; \begin{pmatrix} 6 & 4 \\ 3 & 4 \end{pmatrix}; \begin{pmatrix} 9 & 6 \\ 4 & 4 \end{pmatrix}; \begin{pmatrix} 12 & 8 \\ 5 & 4 \end{pmatrix} \right).$$

3) We then return the codeword

$$c = c_0 + c_1 p \mod p^2 = \left(\begin{pmatrix} 21 & 14 \\ 14 & 22 \end{pmatrix}; \begin{pmatrix} 13 & 3 \\ 22 & 24 \end{pmatrix}; \begin{pmatrix} 5 & 17 \\ 5 & 1 \end{pmatrix}; \begin{pmatrix} 22 & 6 \\ 13 & 3 \end{pmatrix} \right).$$

In this example the error is

$$e = \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 7 & 5 \\ 18 & 8 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}; \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

V. List decoding of generalized Reed-Solomon codes

A. List-decoding over certain valuation rings

In this subsection, as in Subsection IV-A we let A be a ring satisfying $(*)$, S be the set formed by the powers of p, $\mathcal{C} = \mathrm{GRS}_A(v, x, k)$ be a GRS code and G a generator matrix of $\mathcal{C}$ and $\Lambda = A/(p)$ be the residual ring. We precise our black box list decoding algorithm.

Algorithm 4 Black box list decoding algorithm

Input: a received vector y of Λ^n with at most τ errors.
Output: a subset S of $\Lambda^k \times \Lambda^n$ such that $(m, e) \in S \Leftrightarrow mG + e = y$ and $w(e) \leq \tau$.

The list decoding algorithm we propose is recursive and the following algorithm is its recursive step.

Proposition 51. *Algorithm 5 is correct and can decode up to τ errors.*

Proof: The proof is done by descending induction on i. If $i = r - 1$ the proposition holds.

Now let $i < r-1$, $c \in U$, $e = y-c$. There exists $(m_0, e_0) \in S$ such that $c_0 = m_0 G = c \mod p$. Then by Lemma 35 $(c_0 - c)/p \in \mathcal{C}$. Moreover we have $\mathrm{Supp}(e_0) \subseteq \mathrm{Supp}(e)$ and $w(e) \leq \tau$. Therefore $w((e_0 - e)/p) \leq \tau$. We have $y_{c_0} = (y-(c_0+e_0))/p = (c-c_0)/p+(e_0-e)/p$ and by the inductive hypothesis there exists $c_1 \in S_{c_0}$ such that $(c - c_0)/p = c_1 \mod p^{r-(i+1)}$. ∎

The complexity of Algorithm 5 will be studied in detail in Subsection V-C when the ring A is finite. We now give an

Algorithm 5 List decoding from valuation i up to valuation r.

Input: two nonnegative integers $i \leq r$, a received vector y of A^n with at most τ errors. A **black box** list decoding algorithm as specified by Algorithm 4 for the code $\mathcal{C}/p\mathcal{C}$ for decoding up to τ errors.

Output: The set $U \stackrel{def}{=} \{c \in \mathcal{C} : d(c \mod p^{r-i}, y \mod p^{r-i}) \leq \tau\}$.

1: **if** $i = r$ **then**
2: **return** $\{0\}$.
3: **end if**
4: Call the **black box** algorithm with input $(y \mod p)$ to obtain a subset $S \subseteq \Lambda^k \times \Lambda^n$.
5: **for** each $(m_0, e_0) \in S$ **do**
6: $c_0 \leftarrow m_0 G$.
7: Call recursively Algorithm 5 with arguments $i+1$, r and $y_{c_0} = (y - c_0 - e_0)/p$ to get the set S_{c_0} of all the codewords in the ball centered in y_{c_0} of radius τ.
8: **end for**
9: **return** $\{c_0 + c_1 p : c_0 \in S$ and $c_1 \in S_{c_0}$ and $d(c_0 + pc_1 \mod p^{r-i}, y \mod p^{r-i}) \leq \tau\}$.

Algorithm 6 List decoding over a valuation ring.

Input: a positive integer τ, a received vector y of B^n with at most τ errors and a **black box** unique decoding algorithm for $\mathcal{C}/p\mathcal{C}$.

Output: the list of codewords within distance τ of y.

1: $z \leftarrow$ a representative of y in A^n.
2: $z' \leftarrow (v_1^{-1} z_1, \ldots, v_n^{-1} z_n)$.
3: Call Algorithm 5 with parameters 0, r, and z' and obtain the set T.
4: **return** $\{c \mod p^r : c \in T\}$.

algorithm for list decoding a GRS code over $B = A/(p^r)$ for a positive integer r.

Proposition 52. *Algorithm 6 works correctly as expected.*

Proof: This is a direct consequence of Proposition 51 ■

Example 53. In this example we work with the $\mathrm{RS}_{\mathbb{Z}/7^2\mathbb{Z}}(6,2)$ code whose support is $(1,2,3,4,5,6)$. The unique decoding radius is 2 while the list decoding algorithm radius is 3. Suppose we received the word $y = (8,15,22,11,12,13) \in (\mathbb{Z}/7^2\mathbb{Z})^6$. We skip steps 1, 2 and 4 of Algorithm 6 and identify the elements of $\mathbb{Z}_7$ up to precision 2 with the elements of $\mathbb{Z}/7^2\mathbb{Z}$ for the clarity of the example. The execution of Algorithm 6 is as follows:

- We enter Algorithm 5 with $y = (8,15,22,11,12,13)$.
- At step 4, the call to the black box algorithm with $y \mod 7 = (1,1,1,4,5,6)$ returns two codewords and their corresponding errors (step 5):
  1) The codeword $(1,1,1,1,1,1)$ which can be lifted to $(1,1,1,1,1,1)$ and the error $(0,0,0,3,4,5)$.
  2) The codeword $(1,2,3,4,5,6)$ which can be lifted to $(1,2,3,4,5,6)$ and the error $(0,1,2,0,0,0)$.
- We have a list of two candidates, for each one we do a recursive call of Algorithm 5.
- For item 1:
  - We enter *recursively* Algorithm 5 with $[y - (1,1,1,1,1,1) - (0,0,0,3,4,5)]/7 = (1,2,3,1,1,1)$.
  - At step 4 the call to the black box algorithm with $(1,2,3,1,1,1)$ returns the two codewords $(1,1,1,1,1,1,1)$ and $(1,2,3,4,5,6)$ which can be lifted to $(1,1,1,1,1,1,1)$ and $(1,2,3,4,5,6)$ (step 5).
  - At step 9, we return $(1,1,1,1,1,1,1)$ and $(1,2,3,4,5,6)$.
- For item 2:
  - We enter *recursively* Algorithm 5 with $[y - (1,2,3,4,5,6) - (0,1,2,0,0,0)]/7 = (1,1,1,1,1,1)$.
  - At step 4 the call to the black box algorithm with $(1,1,2,1,1,1)$ returns the codeword $(1,1,1,1,1,1,1)$ which can be lifted to $(1,1,1,1,1,1,1)$.
  - At step 9, we return $(1,1,1,1,1,1,1)$.
- Due to the condition of step 9 of Algorithm 5 we return only the two codewords
  - $(1,1,1,1,1,1) + 7 \times (1,2,3,4,5,6) \mod 7^2 = (8,15,22,29,36,43)$ and
  - $(1,2,3,4,5,6) + 7 \times (1,1,1,1,1,1) \mod 7^2 = (8,9,10,11,12,13)$.

  The codeword $(1,1,1,1,1,1) + 7 \times (1,1,1,1,1,1) = (8,8,8,8,8,8)$ is not returned at step 9 because $d(y,(8,8,8,8,8,8)) = 5 > J = 3$.

B. The Guruswami-Sudan algorithm

We now extend the Guruswami-Sudan [26] algorithm to noncommutative GRS codes. We assume in this section that $\{x_1, \ldots, x_n\} \subseteq Z(A)$. Almost nothing has to be changed from the original algorithm. In this subsection we do the following assumption on A: every linear system with coefficients in A with more unknowns than equations has a nonzero solution (with coefficients also in A). This is the case for example when A is any commutative ring or the ring of square matrices over any commutative ring (and therefore any field). We let $J = n - \sqrt{(k-1)n}$.

Lemma 54. *Let $y \in A^n$ be a received word with at most τ errors with $\tau < J$. Then there exists a nonzero bivariate polynomial $Q \in A[X,Y]$ satisfying the three conditions of step 3 of Algorithm 7.*

Proof: As usual we consider the coefficients of Q to be unknowns satisfying the equations $Q(x_i, y_i) = 0$ and $[Q(X + x_i, Y + y_i)]_{s'} = 0$ for $i = 1, \ldots, n$ and $s' = 0, \ldots, s-1$ where $[Q]_{s'}$ denotes the homogeneous component of degree s' of Q.

First note that the value of s in step 2 together with $\tau < J$ imply

$$\left[(n-\tau)^2 - (k-1)n\right] s^2 - (k-1)s - 1 > 0,$$

Algorithm 7 Guruswami-Sudan

Input: a positive integer $\tau < J$ and a received vector y of A^n with at most τ errors.

Output: all the $f \in A[X]_{<k}$ such that $d(y, f(x)) \leq \tau$.

1: $s \leftarrow \left\lfloor \frac{(k-1)n+\sqrt{(k-1)^2n^2+4((n-\tau)^2-(k-1)n)}}{2((n-\tau)^2-(k-1)n)} \right\rfloor + 1$.
2: $L \leftarrow \left\lceil \frac{s(n-\tau)-1}{k-1} \right\rceil - 1$.
3: $y' \leftarrow (v_1^{-1}y_0, \ldots, v_n^{-1}y_n)$.
4: Find $Q = \sum_{i=0}^{L} Q_i(X)Y^i \in (A[X])[Y]$ of degree at most L such that
   1) $Q(x_i, y'_i) = 0$ for all $1 \leq i \leq n$.
   2) $Q(X + x_i, Y + y'_i)$ has valuation at least s.
   3) $\deg Q_i \leq s(n-\tau) - 1 - i(k-1)$ for all $0 \leq i \leq L$.
5: $\mathcal{Z} \leftarrow$ Roots of Q in $A[X]_{<k}$.
6: **return** all the $f \in \mathcal{Z}$ such that $d(y', f(x)) \leq \tau$.

which in turn implies

$$s^2(n-\tau)^2 - 1 > n(k-1)(s^2+s),$$

and then gives

$$\left(\frac{s(n-\tau)-1}{k-1}\right)\left(\frac{s(n-\tau)+1}{2}\right) > n\binom{s+1}{2}.$$

Counting the coefficients of Q we get

$$(L+1)\left(s(n-\tau) - (k-1)\frac{L}{2}\right)$$

unknowns which is greater or equal than

$$\left(\frac{s(n-\tau)-1}{k-1}\right)\left(\frac{s(n-\tau)+1}{2}\right).$$

On the other hand conditions 1 and 2 of step 3 of Algorithm 7 give $n\binom{s+1}{2}$ equations. And we have a nonzero solution by the hypothesis made on A. ∎

Lemma 55. *Let $g \in A[X]$, $Q \in A[X,Y]$ and $z \in Z(A)$. Then*

$$(Q(X, g(X)))(z) = Q(z, g(z)).$$

Proof: According to the definition we took for evaluating polynomials in Subsection IV-B, we have:

$$\begin{aligned}(Q(X,g(X)))(z) &= \sum (Q_{ij}(g(X))^j X^i)(z) \\ &= \sum Q_{ij}(g(z))^j z^i \text{ because } z \in Z(A) \\ &= Q(z, g(z)) \text{ by definition.}\end{aligned}$$

∎

Remark 56. Note that for the Guruswami-Sudan algorithm we could have defined the evaluation of bivariate polynomials in the "usual way" that is, for $f \in A[X]$, $Q \in A[X,Y]$ and $a, b \in A$,

$$Q(a,b) = \sum_{i,j} Q_{ij} a^i b^j$$

and

$$Q(X, f(X)) = \sum_{i,j} Q_{ij} X^i (f(X))^j.$$

As the evaluation is done at points from the center of A both definitions for evaluation give the exact same result.

Lemma 57. *Let $Q \in A[X,Y]$ verifying the three conditions of step 1 of Algorithm 7. Let $f \in A[X]_{<k}$ such that $f(x_i) = y_i$ for a fixed $i \in \{1, \ldots, n\}$. Then $(X - x_i)^s$ divides $Q(X, f(X))$.*

Proof: By assumption we have

$$Q(X + x_i, Y + y_i) = \sum_{\lambda \geq s} \sum_{j+l=\lambda} Q_{jl} X^j Y^l$$

with $Q_{jl} \in A$ and where $s' \geq s$ is the valuation of Q. By Remark 8, there exists a polynomial $g(X) \in A[X]$ such that $f(X) - y_i = g(X)(X - x_i)$. As $x_i \in Z(A)$ we have

$$\begin{aligned}Q(X, f(X)) &= Q((X - x_i) + x_i, (f(X) - y_i) + y_i) \\ &= \sum_{\lambda \geq s'} \sum_{j+l=\lambda} Q_{jl}(g(X)(X - x_i))^l (X - x_i)^j \\ &= \sum_{\lambda \geq s'} \sum_{j+l=\lambda} Q_{jl} g(X)^l (X - x_i)^\lambda \\ &= (X - x_i)^{s'} h(X)\end{aligned}$$

where $h(X) \in A[X]$. ∎

Lemma 58. *Let $Q \in A[X,Y]$ be a bivariate polynomial satisfying the three conditions of step 4 of Algorithm 7 and let $f \in A[X]_{<k}$ be such that $d(y, f(x)) \leq \tau$. Then $Q(X, f(X)) = 0$.*

Proof: Let $f \in A[X]_{<k}$ be a polynomial such that $d(y, f(x)) \leq \tau$. Then $Q(X, f(X))$ is a polynomial of degree at most $s(n-\tau) - 1$. We have $f(x_i) = y_i$ for at least $n - \tau$ values of i. By Lemma 55, $(Q(X, f(X)))(x_i) = Q(x_i, f(x_i)) = 0$ for at least $n - \tau$ values of i.

Denote by E the set $\{i \in \{1, \ldots, n\} : Q(x_i, f(x_i))) = 0\}$ and by $P_r(X)$ the polynomial $\prod_{i \in E}(X - x_i)^r$. We prove by induction on $r \leq s$ that $P_r(X)$ divides $Q(X, f(X))$. For $r = 1$ it is a consequence of Remark 8 and the assumption we made on the support x of the code. By induction there exists $R(X) \in A[X]$ such that $Q(X, f(X)) = R(X)P_r(X)$. Let $i_0 \in E$, by Lemma 57 we also have $Q(X, f(X)) = S(X)(X - x_{i_0})^{r+1}$. By Lemma 7 we have $S(X)(X - x_{i_0}) = R(X)\prod_{i \in E, i \neq i_0}(X - x_i)^r$, whence $R(x_{i_0}) = 0$. This is true for all $i_0 \in E$ and by Remark 8 and the property of the support x we deduce that $P_{r+1}(X)$ divides $Q(X, f(X))$.

It follows that $Q(X, f(X))$ is divisible by a monic polynomial of degree $s(n-\tau)$ which implies $Q(X, f(X)) = 0$. ∎

Proposition 59. *Algorithm 7 works correctly as specified and can correct up to $\lceil J \rceil - 1$ errors.*

Proof: This is direct consequence of Lemma 54 and Lemma 58. ∎

C. Complexities for list decoding algorithms

In order to study the complexity of Algorithm 6, we need a result about the number of codewords that can be returned by Algorithm 7. The results of [32, Section 5] remain valid

r (nilpotency index of $p=3$)	Upper bound
1	0.001310
2	1.386×10^{-6}
3	1.850×10^{-9}
4	2.530×10^{-12}
5	3.469×10^{-15}
6	4.759×10^{-18}
7	6.528×10^{-21}
8	8.954×10^{-24}
9	1.228×10^{-26}
10	1.685×10^{-29}

Fig. 1. Table for $\mathrm{RS}_{\mathrm{GR}(3^r,2)}[8,4,5]$ and a codeword of weight 6.

ℓ (matrix size)	Upper bound
1	0.001310
2	2.530×10^{-12}
3	1.228×10^{-26}
4	1.123×10^{-46}
5	1.930×10^{-72}
6	6.247×10^{-104}
7	3.804×10^{-141}
8	4.358×10^{-184}
9	9.396×10^{-233}
10	3.812×10^{-287}

Fig. 2. Table for $\mathrm{RS}_{M_\ell(\mathbb{F}_9)}[8,4,5]$ and a codeword of weight 6.

in our context. In other words, they do not depend on the algebraic structure of the alphabet. We recall them in the following proposition for the sake of completeness. We assume throughout this subsection that all errors with weight at most τ occur with the same probability regardless of the weight of the transmitted codeword.

Proposition 60. *We let $\mathcal{C} = \mathrm{GRS}_A(n,k)$, $c \in \mathcal{C}$, $w = w(c)$;*

$$N(c,a,i) = 0 \text{ if } w > a + i,$$

and

$$N(c,a,i) = \sum_{j=0}^{\min\left(\left\lfloor \frac{i-(w-a)}{2} \right\rfloor, n-w\right)} \left[\binom{w}{w-a+j} \binom{n-w}{j} (|A|-1)^j \binom{a-j}{i-(w-a)-2j} (|A|-2)^{i-(w-a)-2j} \right]$$

else. We now let

$$M(\tau) = \sum_{u=d}^{\min(2\tau,n)} \sum_{a=u-\tau}^{\tau} \sum_{i=u-a}^{\tau} N(c,a,i).$$

Then the probability that Algorithm 7 returns more than one codeword is at most

$$\frac{M(\tau)}{\sum_{i=0}^{\tau} \left[\binom{n}{i} (|A|-1)^i \right]}.$$

Proof: The proof is identical to [32, Proposition 12, page 9]. ∎

In Tables 1 and 2 we give examples of this upper bound as it is difficult to get a simple asymptotic equivalent. These calculations have been made for finite fields in [32, Section 5] and for Galois rings in [3, Section 5]. As shown in the tables the probability is very small.

Remark 61. As pointed out in the introduction of this subsection the upper bound on the probability given in Proposition 60 is independent of the algebraic structure of the alphabet. Therefore there is no gain in taking the Galois ring or a matrix ring (over a finite field or a Galois ring) instead of the finite field of same cardinality. In fact the advantage resides in the asymptotic complexity of the decoding algorithms given in Subsection IV-A and V-A.

We now let, as in Subsection V-B, $J = n - \sqrt{(k-1)n}$ be the generic Johnson bound. We recall the following proposition:

Proposition 62. *Let $y \in A^n$. Then there exist at most $n(|A|-1)$ codewords within distance J from y.*

Proof: See [25, Corollary 3.3 page 36]. ∎

We can now state the proposition about the complexity of Algorithm 6.

Proposition 63. *With the same notations as in Subsection V-A, let $\mathsf{LDec}(\mathcal{C})$ be the complexity in terms of the number of bit-operations of a list decoding algorithm for $\mathrm{GRS}_{A/(p)}(\mathcal{C})$ and ρ be the probability that the latter algorithm returns more than one codeword. We suppose that $\mathsf{LDec}(\mathcal{C})$ does not depend on ρ. Then Algorithm 6*

- *performs at most*

$$\frac{[n(|A|-1)]^r - 1}{n(|A|-1)-1}(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C})) = (n|A|)^{r-1}(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C})).$$

  bit-operations.
- *performs an* expected *number of at most*

$$\frac{[(1-\rho)+\rho(n(|A|-1))]^r - 1}{[(1-\rho)+\rho(n(|A|-1))]-1}(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C})).$$

  bit-operations.

Proof: It is a direct consequence of Proposition 60 and 62. ∎

Remark 64. Taking the notations of Proposition 63 Recall that $\mathsf{LDec}(\mathcal{C})$ does not depend on ρ. *Heuristically* it is interesting to see that as $\rho \to 0$ we have

$$\frac{[(1-\rho)+\rho(n(|A|-1))]^r - 1}{[(1-\rho)+\rho(n(|A|-1))]-1}(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C})) \to (r + \rho o(1))\,(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C})) = r(\mathsf{LDec}(\mathcal{C}) + \mathsf{Lift}(\mathcal{C}))$$

Therefore the complexity of Algorithm 6 is *heuristically* polynomial in r, n and $|A|$ whenever $\mathsf{LDec}(\mathcal{C})$ and $\mathsf{Lift}(\mathcal{C})$ are polynomial in r, n and $|A|$. This *heuristic* analysis is reasonable according to Tables 1 and 2. Therefore we can notice, as in Proposition 39, that if we denote by B the Galois ring $\mathrm{GR}(p^r, s)$ then the asymptotic complexity given above

is better than the complexity of the corresponding decoding algorithm over the finite field of size p^{rs}.

Corollary 65. *Suppose that A is the Galois ring $\mathrm{GR}(p^r, s)$. Then there exists a list decoding algorithm for $GRS(v, x, k)$ with an asymptotic complexity of*

$$\tilde{O}\left(\frac{n^r(\rho^{rs}-1)^r-1}{n(\rho^{rs}-1)-1} n^7 k^5 s \log p\right) = \tilde{O}(n^{r+6}\rho^{r(r-1)s} k^5 s \log p)$$

arithmetic operations in $\mathbb{F}_p$, or heuristically *an expected number of $\tilde{O}(rn^7k^5s\log p)$ bit-operations which can decode up to the generic Johnson bound.*

Proof: This is a direct consequence of Proposition 63, Remark 64, Lemma 38 and [25, Lemma 6.13, page 111]. ■

Note that the algorithm presented in [25, Algorithm *Poly-Reconstruct*, page 102] applied to a RS code over $\mathbb{F}_{p^{rs}}$ performs at most $\tilde{O}(n^7k^5rs)$ arithmetic operations in $\mathbb{F}_p$. We have an *heuristical* result for list decoding similar to Theorem 3:

Heuristical Result 66. *Given a finite field A, a Galois ring B such that $|A| = |B|$, a GRS code $\mathcal{C}_A$ over A of parameters $[n, k, n-k+1]_A$ and a list decoding algorithm `LDec` for $\mathcal{C}_A$. Suppose that there exists a GRS code $\mathcal{C}_B$ over B of parameters $[n, k, n-k+1]_B$. Then there exists a list decoding algorithm for $\mathcal{C}_B$ with a better asymptotic complexity than `LDec`.*

VI. Conclusion

In this paper we showed that, with strong constraints on their supports, GRS codes can be considered over non commutative rings. But this generalization does not lead to better codes than GRS codes over commutative rings in terms of the parameters.

We also proposed two new decoding algorithms with a low complexity for GRS codes over Galois rings and rings of matrices over a Galois ring. Using these algorithms we showed that given a prime power q and a unique (resp. list) decoding algorithm for a GRS code over $\mathbb{F}_q$ there exists a unique (resp. list) decoding algorithm for a GRS code with same parameters (provided that the GRS code exists with our conditions on its support) over $\mathbb{Z}/q\mathbb{Z}$ with a better asymptotic complexity.

Acknowledgment

The authors would like to thank Daniel Augot for his precious advices and Alain Couvreur and Grégoire Lecerf for their careful readings of this article.

References

[1] M. Alekhnovich, "Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2257–2265, 2005.

[2] M. A. Armand, "Improved list decoding of generalized Reed-Solomon and alternant codes over rings," in *IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 384.

[3] ——, "List decoding of generalized Reed-Solomon codes over commutative rings," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 411–419, 2005.

[4] ——, "Solving the Welch-Berlekamp key equation over a Galois ring," in *WSEAS Transactions on Mathematics*, vol. 4, no. 1, 2005, pp. 319–326.

[5] M. Armand, "Improved list decoding of generalized reed-solomon and alternant codes over galois rings," *IEEE Trans. Inform. Theory*, vol. 51, no. 2, pp. 728–733, feb 2005.

[6] M. Armand and O. de Taisne, "Multistage list decoding of generalized reed-solomon codes over galois rings," *Communications Letters, IEEE*, vol. 9, no. 7, pp. 625–627, jul 2005.

[7] M. Atiyah and I. MacDonald, *Introduction to commutative algebra*, ser. Addison-Wesley series in mathematics. Westview Press, 1994.

[8] D. Augot and A. Zeh, "On the Roth and Ruckenstein Equations for the Guruswami-Sudan Algorithm," in *IEEE International Symposium on Information Theory - ISIT 2008*. Toronto, Canada: IEEE, Jul. 2008, pp. 2620–2624.

[9] N. Babu and K.-H. Zimmermann, "Decoding of linear codes over galois rings," *Information Theory, IEEE Transactions on*, vol. 47, no. 4, pp. 1599–1603, may 2001.

[10] M. Barbier, C. Chabot, and G. Quintin, "On Quasi-Cyclic Codes as a Generalization of Cyclic Codes," *ArXiv:1108.3754*, Aug. 2011.

[11] E. R. Berlekamp, *Algebraic coding theory*, ser. M-6. Aegean Park Press, 1984.

[12] E. R. Berlekamp and L. R. Welch, "Error correction for algebraic block codes," 1986, patent 4633470.

[13] D. Boucher, W. Geiselmann, and F. Ulmer, "Skew-cyclic codes," *Applicable Algebra in Engineering, Communication and Computing*, vol. 18, pp. 379–389, 2007.

[14] D. Boucher, P. Solé, and F. Ulmer, "Skew constacyclic codes over Galois rings," *Advances in mathematics of communications*, vol. 2, no. 3, pp. 273–292, 2008.

[15] D. Boucher and F. Ulmer, "Codes as modules over skew polynomial rings," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Parker, Ed. Springer Berlin / Heidelberg, 2009, vol. 5921, pp. 38–55.

[16] ——, "Coding with skew polynomial rings," *Journal of Symbolic Computation*, vol. 44, no. 12, pp. 1644–1656, 2009.

[17] E. Byrne, "Lifting decoding schemes over a galois ring," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, ser. Lecture Notes in Computer Science, S. Boztas and I. Shparlinski, Eds. Springer Berlin / Heidelberg, 2001, vol. 2227, pp. 323–332.

[18] D. G. Cantor and E. Kaltofen, "On fast multiplication of polynomials over arbitrary algebras," *Acta Inform.*, vol. 28, pp. 693–701, 1991.

[19] L. Chaussade, P. Loidreau, and F. Ulmer, "Skew codes of prescribed distance or rank," *Designs, Codes and Cryptography*, vol. 50, pp. 267–284, 2009.

[20] N. Chen and Z. A Yan, "Complexity analysis of reed-solomon decoding over GF(2^m) without using syndromes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, 2008.

[21] M. Fürer, "Faster Integer Multiplication," in *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*. ACM, 2007, pp. 57–66.

[22] S. Gao, "A new algorithm for decoding Reed-Solomon codes," in *Communications, Information and Network Security, V. Bhargava, H.V. Poor, V. Tarokh, and S. Yoon*. Kluwer, 2002, pp. 55–68.

[23] Gathen, J. von zur and J. Gerhard, *Modern computer algebra*, 2nd ed. Cambridge University Press, 2003.

[24] M. Greferath and U. Vellbinger, "Efficient decoding of $\mathbb{Z}_{p^k}$-linear codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 1288–1291, may 1998.

[25] V. Guruswami, *List decoding of error-correcting codes: winning thesis of the 2002 ACM doctoral dissertation competition*, ser. Lecture Notes in Computer Science. Springer, 2004.

[26] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1757–1767, 1998.

[27] J. Justesen, "On the complexity of decoding Reed-Solomon codes (corresp.)," *IEEE Trans. Inform. Theory*, vol. 22, no. 2, pp. 237–238, Mar. 1976.

[28] R. Kötter, "On Algebraic Decoding of Algebraic-Geometric and Cycling Codes," Ph.D. dissertation, Linköping University, Sweden, 1996.

[29] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, 2003.

[30] S. Lang, *Algebra*, 3rd ed., ser. Graduate Texts in Mathematics. Springer-Verlag, 2002, vol. 211.

[31] J. McConnell, J. Robson, and L. Small, *Noncommutative Noetherian rings*, ser. Graduate studies in mathematics. American Mathematical Society, 2001.

[32] R. R. Nielsen and T. Hoeholdt, "Decoding Reed-Solomon codes beyond half the minimum distance," in *Coding Theory, Cryptography and Related Areas*, J. Buchmann, T. Hoeholdt, H. Stichtenoth, and H. Tapia-Recillas, Eds. Springer-Verlag, Apr. 2000.

[33] G. Norton and A. Salagean-Mandache, "On the key equation over a commutative ring," *Designs, Codes and Cryptography*, vol. 20, pp. 125–141, 2000.

[34] W. Peterson and E. Weldon, *Error-correcting codes*. MIT Press, 1972.

[35] R. Raghavendran, "Finite associative rings," *Compositio Math.*, vol. 21, pp. 195–229, 1969.

[36] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[37] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," in *IEEE Trans. Inform. Theory*, 1998, p. 56.

[38] A. Schönhage and V. Strassen, "Schnelle Multiplikation grosser Zahlen," *Computing*, vol. 7, pp. 281–292, 1971.

[39] J.-P. Serre, *Corps locaux*, ser. Actualités scientifiques et industrielles. Hermann, 1962, no. ns 1296 à 1297.

[40] P. Smith, "Injective modules and prime ideals," *Communications in Algebra*, vol. 9, no. 9, pp. 989–999, 1981.

[41] M. Sudan, "Decoding Reed-Solomon codes beyond the error-correction diameter," in *the 35th Annual Allerton Conference on Communication, Control and Computing*, 1997, pp. 215–224.

[42] T. K. Truong, W. L. Eastman, I. S. Reed, and I. S. Hsu, "Simplified procedure for correcting both errors and erasures of Reed-Solomon code using Euclidean algorithm," *IEEE Proc. Comput. and Digit. Tech.*, vol. 135, no. 6, pp. 318–324, 1988.

[43] S. Wicker and V. Bhargava, *Reed-Solomon Codes and Their Applications*. John Wiley & Sons, 1999.

[44] J. Wood, "Duality for modules over finite rings and applications to coding theory." *American Journal of Mathematics*, vol. 121, no. 3, pp. 555–575, 1999.