HAL
open science

Polynomial Ring Transforms for Efficient XOR-based Erasure Coding

Jonathan Detchart, Jérôme Lacan

To cite this version:

Jonathan Detchart, Jérôme Lacan. Polynomial Ring Transforms for Efficient XOR-based Erasure Coding. IEEE International Symposium on Information Theory (ISIT 2017), Jun 2017, Aachen, Germany. pp. 604-608. hal-01697211

HAL Id: hal-01697211

https://hal.science/hal-01697211v1

Submitted on 31 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive Toulouse Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of some Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in: https://oatao.univ-toulouse.fr/18723

Official URL: http://dx.doi.org/10.1109/ISIT.2017.8006599

To cite this version :

Detchart, Jonathan and Lacan, Jérôme Polynomial Ring Transforms for Efficient XOR-based Erasure Coding. (2017) In: IEEE International Symposium on Information Theory (ISIT 2017), 25 June 2017 - 30 June 2017 (Aachen, Germany).

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Polynomial Ring Transforms for Efficient XOR-based Erasure Coding

Jonathan Detchart, Jérôme Lacan
ISAE-Supaéro, Université de Toulouse, France

***Abstract*—The complexity of software implementations of MDS erasure codes mainly depends on the efficiency of the finite field operations implementation. In this paper, we propose a method to reduce the complexity of the finite field multiplication by using simple transforms between a field and a ring to perform the multiplication in a ring. We show that moving to a ring reduces the complexity of the operations. Then, we show that this construction allows the use of simple scheduling to reduce the number of operations.**

I. Introduction

Most of practical Maximum Distance Separable (MDS) packet erasure codes are implemented in software. In the various applications like packet erasure channels [1] or distributed storage systems [2], the coding/decoding process performs operations over finite fields. The efficiency of the implementation of these finite field operations is thus critical for these applications.

To speedup this operation, [1] described an implementation of finite field multiplications which only uses simple `xor` operations, contrarily to classic software multiplications which are based on lookup tables (LUT). The complexity of multiplying by an element, *i.e.* the number of `xor` operations, depends on the size of the finite field and also on the element itself. This kind of complexity is studied for Maximum-Distance Separable (MDS) codes in [3]. Other work has been done to reduce redundant `xor` operations by applying scheduling [4].

Independently, in the context of large finite field for cryptographic applications, [5] proposed a `xor`-based method to perform fast hardware implementations of multiplications by transforming each element of a field into an element of a larger ring. In this polynomial ring, where the operations on polynomials are done modulo $x^n + 1$, the multiplication by a monomial is much simpler as the modulo is just a cyclic shift. The authors identified two classes of fields based on irreducible polynomials with binary coefficients allowing to transform each field element into a ring element by adding additional "ghost bits".

In this paper, we extend their approach to define fast software implementations of `xor`-based erasure codes. We propose an original method called PYRIT (PolYnomial RIng Transform) to perform operations between elements of a finite field into a bigger ring by using simple transforms between these two structures. Working in such a ring is much easier than working in a finite field. Firstly, it reduces the coding complexity by design. And secondly, it allows the use of simple scheduling to reduce the number of operations thanks to the properties of the ring structure.

The next section presents the algebraic framework allowing to define the various transforms between the finite field and some subsets of the ring. Then we discuss about the choice of these transforms and their properties. We also detail the complexity analysis before introducing some scheduling results.

II. Algebraic context

The algebraic context of this paper is finite fields and ring theory. More detailed presentation of this context including the proofs of the following propositions can be found in [6] or [7].

Definition 1. *Let $\mathbb{F}_{q^w}$ be the finite field with q^w elements.*

Definition 2. *Let $R_{q,n} = \mathbb{F}_q[x]/(x^n - 1)$ denote the quotient ring of polynomials of the polynomial ring $\mathbb{F}_2[x]$ quotiented by the ideal generated by the polynomial $x^n - 1$.*

Definition 3. *Let $p_1^{u_1}(x)p_2^{u_2}(x)\dots p_r^{u_r}(x) = x^n - 1$ be the decomposition of $x^n - 1$ into irreducible polynomials over $\mathbb{F}_q$.*

When n and q are relatively prime, it can be shown that $u_1 = u_2 = \dots = u_n = 1$ (see [7]). In other words, if $q = 2$, and n is odd, we simply have $p_1(x)p_2(x)\dots p_r(x) = x^n - 1$.

In the rest of this document, we assume that n and q are relatively prime.

Proposition 1. *The ring $R_{q,n}$ is equal to the direct sum of its r minimal ideals of $A_i = ((x^n - 1)/p_i(x))$ for $i = 1, \dots, r$.*

Moreover, each minimal ideal contains a unique primitive idempotent $\theta_i(x)$. A construction of this idempotent is given in [6], Chap. 8, Theorem 6.
Since $\mathbb{F}_q[x]/(p_i(x))$ is isomorphic to the finite field $B_i = \mathbb{F}_{q_i^w}$, where $p_i(x)$ is of degree w_i, we have:

Proposition 2. *$R_{q,n}$ is isomorphic to the following Cartesian product:*

$$R_{q,n} \simeq B_1 \otimes B_2 \otimes \dots B_r$$

For each $i = 1, \dots, r$, A_i is isomorphic to B_i. The isomorphism is:

$$\phi_i : \begin{array}{ccc} B_i & \rightarrow & A_i \\ b(x) & \rightarrow & b(x)\theta_i(x) \end{array} \tag{1}$$

and the inverse isomorphism is:

$$\phi_i^{-1} : \begin{array}{ccc} A_i & \rightarrow & B_i \\ a(x) & \rightarrow & a(\alpha_i) \end{array} \tag{2}$$

where α_i is a root of $p_i(x)$.

Let us now assume that $q = 2$. Let us introduce a special class of polynomials:

Definition 4. *The All One Polynomial (AOP) of degree w is defined as*

$$p(x) = x^w + x^{w-1} + x^{w-2} + \dots + x + 1$$

The AOP of degree w is irreducible over $\mathbb{F}_2$ if and only if $w+1$ is a prime and w generates $\mathbb{F}^*_{w+1}$, where $\mathbb{F}^*_{w+1}$ is the multiplicative group in $\mathbb{F}_{w+1}$ [8]. The values $w+1$, such that the AOP of degree w is irreducible is the sequence A001122 in [9]. The first values of this sequence are: $3, 5, 11, 13, 19, 29, \ldots$. In this paper, we only consider irreducible AOP.

According to Proposition 2, $R_{2,w+1}$ is equal to the direct sum of its principal ideals $A_1 = ((x^{w+1}+1)/p(x)) = (x+1)$ and $A_2 = ((x^{w+1}+1)/x+1) = (p(x))$ and $R_{2,w+1}$ is isomorphic to the direct product of $B_1 = \mathbb{F}_2[x]/(p(x)) = \mathbb{F}_{2^w}$ and $B_1 = \mathbb{F}_2[x]/(x+1) = \mathbb{F}_2$.

It can be shown that the primitive idempotent of A_1 is $\theta_1 = p(x)+1$. This idempotents is used to build the isomorphism ϕ_1 between A_1 and B_1.

III. Transforms between the field and the ring

This section presents different transforms between the field $B_1 = \mathbb{F}_{2^w} = \mathbb{F}_2[x]/(p(x))$ and the ring $R_{2,w+1} = \mathbb{F}_2[x]/(x^{w+1}+1)$.

A. Isomorphism transform

The first transform is simply the application of the basic isomorphism between B_1 and the ideal A_1 of $R_{2,w+1}$ (see Prop. 2).

By definition of the isomorphism, we have:

$$\phi_1^{-1}(\phi_1(u(x)).\phi_1(v(x))) = u(x).v(x)$$

So, ϕ_1 can be used to send the elements of the field in the ring, then, to perform the multiplication, and then, to come back in the field. We show in the following Proposition that the isomorphism admits a simplified version.

Let $W(b(x))$, the weight of $b(x)$, defined as the number of monomials in the polynomial representation of $b(x)$.

Proposition 3.

$$\phi_1(b_B(x)) = b_A(x) = \begin{cases} b_B(x) & \textit{if } W(b_B(x)) \textit{ is even} \\ b_B(x)+p(x) & \textit{else} \end{cases}$$

$$\phi_i^{-1}(b_A(x)) = b_B(x) = \begin{cases} b_A(x) & \textit{if } b_w = 0 \\ b_A(x)+p(x) & \textit{else} \end{cases}$$

where b_w is the coefficient of the monomial of degree w of $b_A(x)$.

Proof: For the first point, we have $\phi_1(b(x)) = b(x)\theta_1(x) = b(x)(p(x)+1) = b(x)p(x)+b(x)$. We can observe that $b(x)p(x) = 0$ when $W(b(x))$ is even and $b(x)p(x) = p(x)$ when $W(b(x))$ is odd. The first point is thus obvious.

For the second point, it can be observed that, from the first point of this proposition, if an element of A_1 has a coefficient $b_w \neq 0$, then it was necessarily obtained from the second rule, *i.e.* by adding $p(x)$. Then, its image into B_1 can be obtained by subtracting (adding in binary) $p(x)$. If $b_w = 0$, then nothing has to be done to obtain $b_B(x)$. ■

B. Embedding transform

Let us denote by ϕ_E the embedding function which simply consists in considering the element of the field as an element of the ring without any transformation. This function was initially proposed in [5].

Note that the images of the elements of B_1 doesn't necessarily belong to A_1. However, let us define the function $\bar{\phi}_1^{-1}$ from $R_{2,w+1}$ to A_1 by $\bar{\phi}_1^{-1}(b_A(x)) = b_A(\alpha)$, where α is a root of $p(x)$. This function can be seen as an extension of the function ϕ_1^{-1} to the whole ring.

Proposition 4. *[5] For any $u(x)$ and $v(x)$ in B_1, we have:*

$$\bar{\phi}_1^{-1}(\phi_E(u(x)).\phi_E(v(x))) = u(x).v(x)$$

Proof. The embedding function corresponds to a multiplication by 1 in the ring. In fact, 1 is equal to the sum of the idempotents $\theta_i(x)$ of the ideals A_i, for $i = 1, \ldots, l$ [6, chapter 8, thm. 7]. Thus, $\phi_E(u(x)) = u(x).\sum_{i=0}^{l}\theta_i(x)$. Then, $\phi_E(u(x)).\phi_E(v(x))$ is equal to $u(x).v(x).(\sum_{i=0}^{l}\theta_i(x))^2$. Thanks to the properties of idempotents, $\theta_i(x).\theta_j(x)$ is equal to $\theta_i(x)$ if $i = j$ and 0 else. Thus, $\phi_E(u(x)).\phi_E(v(x))$ is equal to $u(x).v(x).(\sum_{i=0}^{l}\theta_i(x))$. The function $\bar{\phi}_i^{-1}$ is the computation of the remainder modulo $p_i(x)$. The irreducible polynomial $p_i(x)$ corresponds to the ideal A_i. Thus $\theta_i(x)$ mod $p(x)$ is equal to 1 if $i = 1$ and 0 else. □

This proposition proves that the Embedding function can be used to perform a multiplication in the ring instead of doing it in the field. The isomorphism also has this property, but the complexities of the transforms between the field and the ring are more complex.

C. Sparse transform

Let us define the transform ϕ_S from B_1 to $R_{2,w+1}$:

$$\phi_S(b_B(x)) = b_A(x) = \phi_1(b_B(x)) + \delta.p(x)$$

where $\delta = 1$ if $W(\phi_1(b_B(x))+p(x)) < W(\phi_1(b_B(x)))$ and 0 else.

Proposition 5. *For any $u(x)$ and $v(x)$ in B_1, we have:*

$$\bar{\phi}_1^{-1}(\phi_S(u(x)).\phi_S(v(x))) = u(x).v(x)$$

Proof. As observed in the proof of Prop. 4, $\bar{\phi}_i^{-1}$ is just the computation of the remainder modulo $p(x)$. Moreover, according to the definition of ϕ_S, $\phi_S(u(x)).\phi_S(v(x))$ is equal to $u(x).v(x)$ plus a multiple of $p(x)$ (possibly equal to 0). Thus, the remainder of $\phi_S(u(x)).\phi_S(v(x))$ modulo $p(x)$ is equal to $u(x).v(x)$. □

This proposition shows that ϕ_S can be used to perform the multiplication in the ring. The main interest of this transform is that the weight of the image of ϕ_S is small, which reduce the complexity of the multiplication in the ring.

D. Parity transform

Proposition 6. *The ideal A_1 is composed of the set of elements of $R_{2,w+1}$ with even weight.*

Proof: We can observe from Proposition 3 that all the image of ϕ_1 have even weight. Since the number of even-weight element of $R_{2,w+1}$ is equal to the number of elements of A_1, A_1 is composed of the set of elements of $R_{2,w+1}$ with even weight. ■

Let us consider the function ϕ_P, from B_1 to $R_{2,w+1}$, which adds a single parity bit to the vector corresponding to the finite field element. The obtained element has an even weight (by construction), and thus, according to the previous Proposition, it belongs to A_1.

Since the images by ϕ_P of two distinct elements are distinct, ϕ_P is a bijection between B_1 and A_1. The inverse function, ϕ_P^{-1}, consists just in removing the last coefficient of the ring element.

It should be noted that ϕ_P is not an isomorphism, but just a bijection between B_1 and A_1. However it will be shown in next Section that this function can be used in the context of erasure codes.

IV. Application of transforms

In typical `xor`-based erasure coding systems [1], the encoding process consists in multiplying an information vector by the generator matrix. Since in software, `xor` are performed using machine words of l bits, l interleaved codewords are encoded in parallel.

We consider a system with k input data blocks and m output parity blocks.

The total number of `xor` of the encoding is thus defined by the generator matrix which must be as sparse as possible. First, we use a $k \times (k+m)$-systematic generator matrix built from a $k \times k$-identity matrix concatenated to a $k \times m$ Generalized Cauchy (GC) matrix [10]. A GC matrix generates a systematic MDS code and it contains only 1 on its first row and on its first column. Then, to improve the sparsity of the generator matrix in the ring, we use the Sparse transform ϕ_S. This has to be done only once since the ring matrix is the same for all the codewords.

For the information vectors, it is not efficient to use ϕ_S since the `xor`s of machine words do not take into account the sparsity of the `xor`-ed vectors. We thus use Embedding or Parity transforms, which are less complex than ϕ_1.

When Embedding is used for information vectors and Sparse is used for the generator matrix, the obtained result in the ring can be sent into the field by using ϕ_1^{-1} (proof similar to the proofs of Propositions 4 or 5).

When Parity is used for the information vector, the image of the vector in the ring only contains elements of the ideal A_1. Since these elements are multiplied by the generator matrix (in the ring), the obtained result only contains elements of the ideal A_1. These elements have even weight, so it is not necessary to keep the parity bit before sending them on the "erasure channel". Since Parity transform is not an isomorphism, these data can not be decoded by another method. Indeed, to decode, it is necessary to apply ϕ_P (add the parity bit), then to decode by multiplying by the inverse matrix, and then to to apply ϕ_P^{-1} (remove the parity bit on the correct information vector).

V. Complexity analysis

In this section, we determine the total number of `xor` operations done in the coding and the decoding processes.

A. Coding complexity

The coding process is composed of three phases: the field to ring transform, the matrix vector multiplication and the ring to field transform. We assume that the information vector is a vector of k elements of the field $\mathbb{F}_{2^w}$.

For the first and the third phases, Table I gives the complexities of Embedding and Parity transforms obtained from their definition in Section III.

	field to ring	ring to field
Embedding	0	$m.w$
Parity bits	$k.w$	0

TABLE I
Number of xor for Embedding and Parity transforms

The choice between the two methods thus depends on the values of the parameters: if $k > m$, Parity transform has lower complexity. Else, "Embedding" complexity is better.

For the matrix vector operation, let us first consider the multiplication of two ring elements. As explained in the previous section, the first element (which corresponds to an information symbol) is managed by the software implementation by machine words. So the complexity of the multiplication only depends on the weight of the second element, denoted by $w_2 \in \{0, 1, \ldots, w+1\}$. The complexity of this multiplication is thus $(w+1).w_2$.

Now, we can consider the specificities of the various transforms. In the Parity transform, the last bit of the parity blocks is not used (i.e. it is not transmitted on the erasure channel). So it is not necessary to compute it. It follows that the complexity of the multiplication is only $w.w_2$.

Similarly, for the Embedding transform, the last bit of the input vector is always equal to 0. So, we also have a complexity equal to $w.w_2$.

To have an average number of operations done in the multiplication of the generator matrix by the input data blocks, we have to evaluate the average weight of the entries of the generator matrix in the ring.

The generator matrix is a $k \times m$-GC matrix with the first column and the first row are filled by 1. The other elements can be considered as random non zero elements. They are generated by ϕ_S which chooses the lowest ring element among the two ones corresponding to the field element. Let us denote their average weight by w_{ϕ_S}. For this case, the average number of `xor`s is thus:

$$(k+m-1).w + (k-1).(m-1).w.w_{\phi_S}$$

This leads to the following general expression of the coding complexity:

$$(\min(k,m) + k + m - 1).w + (k-1).(m-1).w.w_{\phi_S}$$

To estimate the complexity on a practical example, we fix the value of w to 4. Classic combinatorial evaluation (not presented here) gives the average weight for nonzeros images of ϕ_S:

$$w_{\phi_S} = \frac{w+1}{2^{w+1}-2}.\left(2^w - \binom{w}{w/2}\right)$$

So, $w_{\phi_S} = 1.66$. We plot in Figure 1 the evolution of the factor over optimal (used *e.g.* in [2], table III) which is the density of the matrix normalized by the minimal density, $k.m.w$. We vary the value of k for three values of m: 3, 5 and 7. For each pair (k, m), we generate 10000 random GC matrices and keep the best we found.

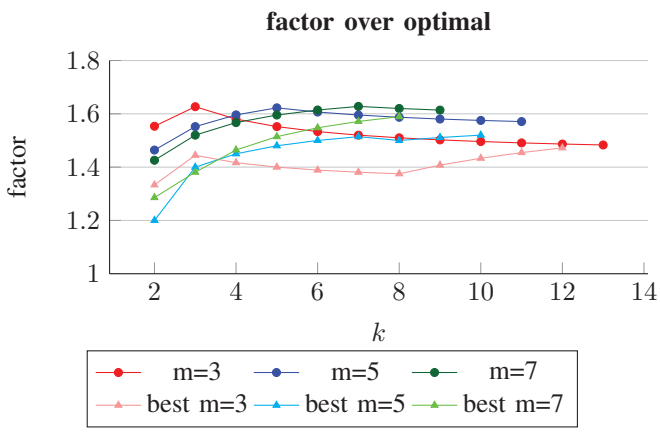

Fig. 1. Factor over optimal depending on m

We can observe that the values are very low. For example, [2] gives the lowest density of Cauchy matrices for the field $\mathbb{F}_{2^4}$ and we can observe that our values are always lower than these ones.

To reduce the complexity in specific cases, we can observe that the ring contains w elements whose the corresponding matrix is optimal (one diagonal). By using these elements, we can search by brute force MDS matrices built only with these optimal elements. For example, let us consider the elements of the field $\mathbb{F}_{2^4}$ sent into the ring $R_{2,5}$. The Vandermonde matrix defined by:

$$V = \left(x^{i.j}\right)_{i=0,\ldots,4;j=0,\ldots,4}$$

where x is a monomial in $R_{2,5}$ has the minimal number of 1. It can be verified that this matrix can be used to build a systematic a MDS code. For this matrix, the total number of `xor`s done in the generation of the parity packets (including the field-to-ring and ring-to-field transforms) is

$$(\min(k,m)+k+m-1).w+(k-1).(m-1).w = k.w+k.m.w$$

Its factor over optimal is equal to 1.2 which is lower than the values given in Figure 1 and which is close to the lowest bound given in [3].

B. Decoding complexity

As the decoding is a matrix inversion and a matrix vector multiplication, we can use the same approach to perform the multiplication. We first invert the sub-matrix in the field, then we transform each entry of this matrix into ring elements. Then, we perform the ring multiplication.

The complexity of the decoding thus depends on the complexity of the matrix inversion and on the complexity of the matrix vector multiplication.

The complexity of the matrix vector multiplication was studied in the previous paragraph.

The complexity of the a $r \times r$-matrix inversion is generally in $O(r^3)$ operations in the field. But if the matrix has a Cauchy structure, this complexity can be reduced to $O(r^2)$ [1].

Note that, contrarily to the matrix vector multiplication, the matrix inversion complexity does not depend on the size of the source and parity blocks. And thus, it becomes negligible when the size of the blocks increase.

VI. Scheduling

An interesting optimization on MDS erasure codes under `xor`-based representation is the scheduling of `xor` operations.

Such techniques are proposed in [11], [12], [2] and [13]. The general principle consists in "factorizing" some `xor` operations which are done several times to generate the parity blocks.

We show in the two next paragraphs that these techniques can be used very efficiently on the ring elements.

However, it can be observed that the matrices defined over rings have two main advantages.

A. Complexity reduction

Over finite fields, the scheduling consists in searching common patterns on the binary representation of the generator matrices. The $w \times w$-matrices representing the multiplication by the field elements does not have particular structure and thus, they must be entirely considered in the scheduling algorithm.

This is not the case for the $(w+1) \times (w+1)$-matrices corresponding to a ring element because, thanks to the form of the polynomial $x^{w+1}+1$, they are composed of diagonals either full of 0 or 1. This means that they can be represented in the scheduling algorithm just by their first column or, equivalently, by the ring polynomial.

This allows to drastically reduce the algorithm complexity and thus to handle bigger matrices. From a polynomial point of view, the search of scheduling just consist in finding some common patterns in the equations generating the parity blocks.

Example . Let us assume that $n = 5$ and that three data polynomials $a_0(x)$, $a_1(x)$ and $a_2(x)$ are combined to generate the three parities $p_0(x) = (1+x^4)a_0(x)+x^2a_1(x)+x^3a_2(x)$, $p_1(x) = a_0(x) + x^3a_1(x) + (1+x^3)a_2(x)$ and $p_2(x) = a_0(x) + a_1(x) + x^3a_2(x)$.

In this case, the scheduling just consists in computing $p'(x) = a_0(x) + x^3a_2(x)$ and then $p_0(x) = p'(x) + x^4a_0(x) + x^2a_1(x)$, $p_1(x) = p'(x) + x^3a_1(x) + a_2(x)$ and $p_2 = p'(x) + a_2(x)$.

To estimate the complexity, we can consider the number of sums of polynomials. Without scheduling, we need 11 sums (4 for $p_0(x)$, 4 for $p_1(x)$, and 3 for $p_2(x)$) instead of with scheduling, we only need 10 sums (2 for $p'(x)$, 3 for $p_0(x)$, 3 for $p_1(x)$ and 2 for $p_2(x)$). ▪

B. Additional patterns

Ring-based matrices allow to find more common patterns than field-based matrices. The main idea is to observe that, in the ring, we can "factorize" not only common operations, but also operations which are multiple by a monomial (*i. e.* cyclic-shift) of operations done in some other equations. This is possible only because the multiplications are done modulo $x^{w+1}+1$.

Example . Let us assume that $n = 5$ and that three data polynomials $a_0(x)$, $a_1(x)$ and $a_2(x)$ are combined to generate the parities $p_0(x) = a_0(x) + x^2a_1(x) + (1 + x^2)a_2(x)$, $p_1(x) = x^2a_0(x) + x^3a_1(x) + (x + x^4)a_2(x)$ and $p_2(x) = x^2a_0(x) + a_1(x) + (x^2 + x^3)a_2(x)$. We can observe that, with a "simple" scheduling, it is not possible to factorize some operations.

However, by rewriting the polynomials, we can reveal factorizations: $p_1(x) = x^2a_0(x) + x(x^2a_1(x) + a_2(x)) + x^4a_2(x)$ and $p_2(x) = x^2a_0(x) + x^3(x^2a_1(x) + a_2(x)) + x^2a_2(x)$. So, if $p'(x) = x^2a_1(x) + a_2(x)$, we have $p_0(x) = p'(x) + a_0(x) + x^2a_2(x)$, $p_1(x) = xp'(x) + x^2a_0(x) + x^2a_2(x)$ and $p_2(x) = x^3p'(x) + x^2a_0(x) + x^2a_2(x)$.

To estimate the complexity by the same method than in the previous example, we need 11 polynomial additions with scheduling compared to 12 additions necessary without scheduling. ▪

C. Scheduling results

To evaluate the potential gain of the scheduling, we have implemented an exhaustive search of the best patterns on generator matrices.

This algorithm was applied on several codes for the field $\mathbb{F}_{2^4}$. Table II presents the results in term of "factor over optimal" which is defined as the total number of 1 in the matrix over the number of 1 for the optimal MDS matrix , i.e $k.m.w$.

When working in a ring, we include to the complexity the operations needed to apply the transforms. In this case, Embedding transform has a lower complexity. So we added $m.w$ to the number of 1 in the matrix resulting from the scheduling algorithm.

For each case, we have generated 100 random Generalized Cauchy matrices.

The measured parameters are:

- average field matrix: average number of 1 in the GC matrices divided by $k.m.w$
- best field matrix: lowest number of 1 among the GC matrices divided by $k.m.w$
- average ring matrix: average number of 1 in the ring matrices (without scheduling) + ring-field correspondence divided by $k.m.w$
- best ring matrix: best number of 1 among the ring matrices (without scheduling) + ring-field correspondence divided by $k.m.w$
- average with scheduling: average number of `xors` with scheduling + ring-field correspondence divided by $k.m.w$
- best with scheduling: best number of `xors` with scheduling + ring-field correspondence divided by $k.m.w$

k+m,k	12,8	16,10
average field matrix	1.79	1.90
best field matrix	1.73	1.85
average ring matrix	1.59	1.63
best ring matrix	1.5	1.58
average with scheduling	1.32	1.26
best with scheduling	1.19	1.20

TABLE II
FACTOR OVER OPTIMAL FOR $w = 4$

This table confirms that, even without scheduling, ring matrices have a lower density than field matrices, thanks to the Sparse transform. Applying scheduling to these matrices allows a significant gain of complexity. Indeed, it reduces the complexity by more than 20% on the best matrices. The final results are similar to the results obtained (without scheduling) on the optimal matrix in Section V-A. To the best of our knowledge, other scheduling approaches do not reach this level of sparsity for these parameters.

VII. Conclusion

In this paper, we have presented a new method to build MDS erasure codes with low complexity. By using transforms between a finite field and a polynomial ring, sparse generator matrices can be obtained. This allows to significantly reduce the complexity of the matrix vector multiplication. It also allows simple schedulers that drastically improve the complexity by reducing the number of operations.

Similar results can be obtained with Equally-Spaced Polynomials (ESP) [5], but they are not presented here due to lack of space.

References

[1] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," in *Technical Report ICSI TR-95-048*, August 1995.

[2] J. S. Plank, "Xor's, lower bounds and mds codes for storage," in *2011 IEEE Information Theory Workshop*, Oct 2011, pp. 503–507.

[3] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Trans. Information Theory*, vol. 45, no. 1, pp. 46–59, 1999.

[4] J. Luo, M. Shrestha, L. Xu, and J. S. Plank, "Efficient encoding schedules for xor-based erasure codes," *IEEE Transactions on Computers*, vol. 63, no. 9, pp. 2259–2272, Sept 2014.

[5] T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of fields $gf(2^m)$," *Information and Computation*, vol. 83, no. 1, pp. 21 – 40, 1989.

[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[7] A. Poli and L. Huguet, *Error correcting codes: theory and applications*. Prentice Hall, 1992. [Online]. Available: https://books.google.fr/books?id=cuxQAAAAMAAJ

[8] P. Wah and M. Wang, "Realization and application of the massey-omura lock," in *Proceedings, International Zurich Seminar*, 1984, pp. 175–185.

[9] N. J. A. Sloane, "A001122 sequence," in *https://oeis.org/A001122*.

[10] R. M. Roth and G. Seroussi, "On generator matrices of mds codes (corresp.)," *IEEE Transactions on Information Theory*, vol. 31, no. 6, pp. 826–830, November 1985.

[11] J. L. Hafner, V. Deenadhayalan, K. K. Rao, and J. A. Tomlin, "Matrix methods for lost data reconstruction in erasure codes," in *Proceedings of the 4th USENIX Conf. on File and Storage Technologies - Volume 4*, ser. FAST'05.

[12] C. Huang, J. Li, and M. Chen, "On optimizing xor-based codes for fault-tolerant storage applications," in *Information Theory Workshop, 2007. ITW '07. IEEE*, Sept 2007, pp. 218–223.

[13] J. S. Plank, C. D. Schuman, and B. D. Robison, "Heuristics for optimizing matrix-based erasure codes for fault-tolerant storage systems," 2012, pp. 1–12.