entropy

MDPI

Article

On Matrix Representation of Extension Field GF(p^L) and Its Application in Vector Linear Network Coding

Hanqi Tang , Heping Liu, Sheng Jin, Wenli Liu and Qifu Sun *

School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China; tanghanqi1009@ustb.edu.cn (H.T.); 13121778655@163.com (H.L.); 13735677295@163.com (S.J.); lilyaccount00@163.com (W.L.)
* Correspondence: qfsun@ustb.edu.cn

Abstract: For a finite field GF(p^L) with prime p and $L > 1$, one of the standard representations is $L \times L$ matrices over GF(p) so that the arithmetic of GF(p^L) can be realized by the arithmetic among these matrices over GF(p). Based on the matrix representation of GF(p^L), a conventional linear network coding scheme over GF(p^L) can be transformed to an L-dimensional vector LNC scheme over GF(p). Recently, a few real implementations of coding schemes over GF(2^L), such as the Reed–Solomon (RS) codes in the ISA-L library and the Cauchy-RS codes in the Longhair library, are built upon the classical result to achieve matrix representation, which focuses more on the structure of every individual matrix but does not shed light on the inherent correlation among matrices which corresponds to different elements. In this paper, we first generalize this classical result from over GF(2^L) to over GF(p^L) and paraphrase it from the perspective of matrices with different powers to make the inherent correlation among these matrices more transparent. Moreover, motivated by this correlation, we can devise a lookup table to pre-store the matrix representation with a smaller size than the one utilized in current implementations. In addition, this correlation also implies useful theoretical results which can be adopted to further demonstrate the advantages of binary matrix representation in vector LNC. In the following part of this paper, we focus on the study of vector LNC and investigate the applications of matrix representation related to the aspects of random and deterministic vector LNC.

Keywords: vector linear network coding; matrix representation; finite field

check for updates

Citation: Tang, H.; Liu, H.; Jin, S.; Liu, W.; Sun, Q. On Matrix Representation of Extension Field GF(p^L) and Its Application in Vector Linear Network Coding. *Entropy* **2024**, *26*, 822. https://doi.org/10.3390/e26100822

Academic Editors: Boris Ryabko and Jun Chen

Received: 30 July 2024
Revised: 8 September 2024
Accepted: 24 September 2024
Published: 26 September 2024

1. Introduction

The finite fields GF(p^L) with a prime of p and an integer of $L \geq 1$ have been widely used in modern information coding, information processing, cryptography, and so on. Specifically, in the study of *linear network coding* (LNC), conventional LNC [1] transmits data symbols along the edges over GF(p^L), and every outgoing edge of a node v transmits a data symbol that is a GF(p^L)-linear combination of the incoming data symbols to v. A general LNC framework called *vector* LNC [2] models the *data unit* transmitted along every edge as an L-dimensional vector of data symbols over GF(p). Correspondingly, the coding operations at v involve GF(p)-linear combinations of all data symbols in incoming data unit vectors and are naturally represented by $L \times L$ matrices over GF(p).

Recently, many works [3–7] have shown that vector LNC has the potential to reduce extra coding overheads in networks relative to conventional LNC. In order to achieve vector LNC, a *matrix representation* of GF(p^L) [8] is $L \times L$ matrices over GF(p) so that the arithmetic of GF(p^L) can be realized by the arithmetic among these matrices over GF(p). Based on the matrix representation of GF(p^L), a conventional LNC scheme over GF(p^L) can be transformed to an L-dimensional vector LNC scheme over GF(p). In addition to the theory of LNC, many existing implementations of linear codes, such as the Cauchy-RS codes in the Longhair library [9] and the RS codes in the Jerasure library [10,11] and the latest release of the ISA-L library [12], also practically achieve arithmetic over GF(2^L) using matrix representation.

In order to achieve the matrix representation of GF(p^L), a classical result obtained in [13] relies on polynomial multiplications to describe the corresponding matrix of an element over GF(2^L). A number of current implementations and studies (see, e.g., [9–15]) utilize such a characterization to achieve the matrix representation of GF(2^L). However, the characterization in the present form focuses more on the structure of every individual matrix and does not shed light on the inherent correlation between matrices that corresponds to different elements. As a result, in the aforementioned existing implementations, the corresponding binary matrix is either independently computed on demand or fully stored in a lookup table as an $L \times L$ matrix over GF(2) in advance.

In the first part of this paper, we shall generalize the characterization of matrix representation from over GF(2^L) to over GF(p^L) and paraphrase it from the perspective of matrices with different powers so that the inherent correlation among these matrices will become more transparent. More importantly, this correlation motivates us to devise a lookup table to pre-store the matrix representation with a smaller size. Specifically, compared to the one adopted in the latest release of the ISA-L library [12], the table size is reduced by a factor of $1/L$. Additionally, this correlation also implies useful theoretical results that can be adopted to further demonstrate the advantages of binary matrix representation in vector LNC. In the second part, we focus on the study of vector LNC and show the applications of matrix representation related to the aspects of random and deterministic coding. In random coding, we theoretically analyze the coding complexity of conventional and vector LNC via matrix representation under the same alphabet size 2^L. The comparison results show that vector LNC via matrix representation can reduce at least half of the coding complexity to achieve multiplications. Then, in deterministic LNC, we focus on the special choice of coding operations that can be efficiently implemented. In particular, we illustrate that the choice of primitive polynomial can influence the distributions of matrices with different numbers of non-zero entries and propose an algorithm to obtain a set of sparse matrices that can be good candidates for the coefficients of a practical LNC scheme.

This paper is structured as follows. Section 2 reviews the mathematical fundamentals of representations to an extension field GF(p^L). Section 3 paraphrases the matrix representation from the perspective of matrices in different powers and then devises a lookup table to pre-store the matrix representation with a smaller size. Section 4 focuses on the study of vector LNC and shows the applications of matrix representation related to the aspects of random and deterministic coding. Section 5 summarizes this paper.

Notation. In this paper, every bold symbol represents a vector or a matrix. In particular, $\mathbf{I}_L$ refers to the identity matrix of size L, and $\mathbf{0}$, $\mathbf{1}$, respectively, represent an all-zero or all-one matrix, whose size, if not explicitly explained, can be inferred in the context.

2. Preliminaries

In this section, we review three different approaches to express an extension field GF(p^L) with p^L elements, where p is a prime. The first approach is the standard *polynomial representation*. Let $p(x)$ denote an irreducible polynomial of degree L over GF(p) and α be a root of $p(x)$. Every element of GF(p^L) can be uniquely expressed as a polynomial in α over GF(p) with a degree less than L, and $\{1, \alpha, \alpha^2, \ldots, \alpha^{L-1}\}$ forms a basis GF(p^L) over GF(p). In particular, every $\beta \in \mathrm{GF}(p^L)$ can be uniquely represented in the form of $\sum_{l=0}^{L-1} v_l \alpha^l$ with $v_l \in \mathrm{GF}(p)$. In the polynomial representation, the element $\beta = \sum_{l=0}^{L-1} v_l \alpha^l$ is expressed as the L-dimensional *representative vector* $\mathbf{v}_\beta = [v_0\, v_1\, \ldots\, v_{L-1}]_\beta$ over GF(p). In order to further simplify this expression, $\mathbf{v}_\beta$ can be written as the integer $0 \le d_\beta^{\mathrm{poly}} \le p^L - 1$ such that

$$d_\beta^{\mathrm{poly}} = \sum_{l=0}^{L-1} p^l \hat{v}_l, \tag{1}$$

where $0 \le \hat{v}_l < p$ is the integer representation of v_l, that is, $\sum_{i=1}^{\hat{v}_l} 1 = v_l$ where 1 is to be the multiplicative unit of GF(p).

The second approach is called the *generator representation*, which further requires $p(x)$ to be a primitive polynomial such that α is a primitive element, and all $p^L - 1$ non-zero elements in GF(p^L) can be generated as $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p^L-2}$. Thus, every non-zero $\beta \in \mathrm{GF}(p^L)\backslash 0$ is uniquely expressed as the integer $0 \le d_\beta^{\mathrm{gen}} \le p^L - 2$ subject to

$$\beta = \alpha^{d_\beta^{\mathrm{gen}}}. \tag{2}$$

The *polynomial representation* clearly specifies the additive structure of GF(p^L) as a vector space or a quotient ring of polynomials over GF(p) while leaving the multiplicative structure hard to determine. Meanwhile, the *generator representation* explicitly illustrates the cyclic multiplicative group structure of $\mathrm{GF}(p^L)\backslash\{0\}$ without clearly demonstrating the additive structure. It turns out that the addition operation and its inverse in GF(p^L) are easy to implement based on the *polynomial representation*, while the multiplicative operations and its inverse in GF(p^L) are easy to be implement based on the *generator representation*. In particular, for $\beta_1, \beta_2 \in \mathrm{GF}(p^L)$,

$$d_{\beta_1+\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} \oplus d_{\beta_2}^{\mathrm{poly}} \text{ or equivalently } \mathbf{v}_{\beta_1+\beta_2} = \mathbf{v}_{\beta_1} + \mathbf{v}_{\beta_2} \tag{3}$$

$$d_{\beta_1\beta_2}^{\mathrm{gen}} = (d_{\beta_1}^{\mathrm{gen}} + d_{\beta_2}^{\mathrm{gen}}) \mod p^L - 1, \tag{4}$$

where the operation $\oplus$ between two integers $d_{\beta_1}^{\mathrm{poly}}$ and $d_{\beta_2}^{\mathrm{poly}}$ means the component-wise p-ary addition $\mathbf{v}_1 + \mathbf{v}_2$ between the p-ary expression $\mathbf{v}_1$, $\mathbf{v}_2$ of them. This is the key reason that in practice both representations are always adopted interchangeably when conducting operations in GF(p^L).

Unfortunately, except for some special $\beta \in \mathrm{GF}(p^L)$, such as $\alpha^l, 0 \le l < L$, there is not a straightforward way to establish the mapping between d_β^{poly} and d_β^{gen} without computation, and a built-in lookup table is always adopted in practice to establish the mapping between two types of representations. For instance, Table 1 lists the mapping between d_β^{poly} and d_β^{gen} for non-zero elements β in GF(2^3) with $p(x) = x^4 + x + 1$.

Table 1. The mapping between d_β^{poly} and d_β^{gen} for non-zero β in GF(2^4) with $p(x) = x^4 + x + 1$.

d_β^{gen}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d_β^{poly}	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9

By convention, elements β in GF(p^L) are represented as d_β^{poly}. It takes L p-ary additions to compute $d_{\beta_1+\beta_2} = d_{\beta_1}^{\mathrm{poly}} \oplus d_{\beta_1}^{\mathrm{poly}}$. Based on the lookup table, it takes 3 lookups (which, respectively, map $d_{\beta_1}^{\mathrm{poly}}, d_{\beta_2}^{\mathrm{poly}}$ to $d_{\beta_1}^{\mathrm{gen}}, d_{\beta_2}^{\mathrm{gen}}$ and $d_{\beta_1\beta_2}^{\mathrm{gen}}$ to $d_{\beta_1\beta_2}^{\mathrm{poly}}$), 1 integer addition, and at most 1 modulo $p^L - 1$ operation to compute $d_{\beta_1\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} d_{\beta_2}^{\mathrm{poly}}$. Meanwhile, it is worthwhile to note that the calculation of $d_{\beta_1\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} d_{\beta_2}^{\mathrm{poly}}$ without the table follows the multiplication of polynomials $f_1(x)$ and $f_2(x)$ with coefficient vectors $\mathbf{v}_{\beta_1}$ and $\mathbf{v}_{\beta_2}$, respectively, and finally falls into

$$f_1(x)f_2(x) \text{ modulo } p(x), \tag{5}$$

where the computational complexity compared with the following matrix representation will be fully discussed in Section 4.

The third approach, which is the focus of this paper, is given by means of matrices called the *matrix representation* [8]. Let $\mathbf{C}$ be the $L \times L$ companion matrix of an irreducible polynomial $p(x)$ of degree L over GF(p). In particular, if $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{L-1}x^{L-1} + x^L$ with $a_0, a_1, \ldots, a_{L-1} \in \mathrm{GF}(p)$,

$$\mathbf{C} = \begin{bmatrix} \mathbf{0} & -a_0 \\ & -a_1 \\ \mathbf{I}_{L-1} & \cdots \\ & -a_{L-1} \end{bmatrix}_{L\times L}. \tag{6}$$

It can be easily verified that $p(x)$ is the characteristic polynomial of $\mathbf{C}$, and according to the Cayley–Hamilton theorem, $p(\mathbf{C}) = \mathbf{0}$. As a result, $\{\mathbf{I}_L, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{L-1}\}$ forms a basis of GF(p^L) over GF(p), and for every $\beta \in \mathrm{GF}(p^L)$ with the representative vector $\mathbf{v}_\beta = [v_0\ v_1\ \ldots\ v_{L-1}]^{\mathrm{T}}$ based on the polynomial representation, the matrix representation $\mathbf{M}(\beta)$ of β is defined as

$$\mathbf{M}(\beta) = \textstyle\sum_{i=0}^{L-1} v_i \mathbf{C}^i. \tag{7}$$

If the considered $p(x)$ further qualifies as a primitive polynomial, then similar to the role of the primitive element α defined above, $\mathbf{C}$ is also a multiplicative generator of all non-zero elements in GF(p^L), that is, $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ for all $0 \le i \le p^L - 2$. One advantage for the matrix representation is that all operations in GF(p^L) can be realized by matrix operations over GF(p) among the matrices in $\mathcal{C}$ such that there is no need to interchange between the polynomial and the generator representations in performing field operations. For more detailed discussions of representation of an extension field, please refer to [16].

Based on the polynomial representation and generator representation, even though the arithmetic over GF(p^L) can be efficiently realized by (3), (4) and a lookup table, it requires two different types of calculation systems, i.e., one over GF(p) and the other over integers. This hinders the deployment practicality in applications with resource-constrained edge devices, such as in ad hoc networks or Internet of Things applications. In comparison, the matrix representation of GF(p^L) interprets the arithmetic of GF(p^L) solely over the arithmetic over GF(p), so it is also a good candidate for realization of the efficient implementation of linear codes over GF(p^L) such as in [9–13].

3. Useful Characterization of the Matrix Representation

Let $p(x)$ be a defined irreducible polynomial over GF(p) of degree L and let $\alpha \in \mathrm{GF}(p^L)$ be a root of $p(x)$. When $p = 2$, a useful characterization of the matrix representation $\mathbf{M}(\beta)$ of $\beta \in \mathrm{GF}(p^L)$ (with respect to $p(x)$) can be deduced based on the following classical result obtained in Construction 4.1 and Lemma 4.2 of [13]: For $1 \le j \le L$, the jth column in $\mathbf{M}(\beta)$ is equal to the binary expression of $\alpha^{j-1}\beta$ based on the polynomial representation. A number of implementations and studies (see, e.g., [9–15]) of linear codes utilize such characterization to achieve the matrix representation of GF(2^L). However, the characterization in the present form relies on polynomial multiplications and focuses more on the structure of every individual $\mathbf{M}(\beta)$. It does not explicitly shed light on the inherent correlation among $\mathbf{M}(\beta)$ of different $\beta \in \mathrm{GF}(2^L)$. It turns out that in existing implementations, such as the Cauchy-RS codes in the Longhair library [9] and the RS codes in the Jerasure library [10,11], and the latest release of ISA-L library [12], $\mathbf{M}(\beta)$ is either independently computed on demand or fully stored in a lookup table as an $L \times L$ matrix over GF(2) in advance.

In this section, we shall generalize the characterization of matrix representation from over GF(2^L) to over GF(p^L) and paraphrase it based on the interplay with the generator representation instead of the conventional polynomial representation so that the correlation among $\mathbf{M}(\beta)$ of different $\beta \in \mathrm{GF}(p^L)$ will become more transparent. From now on, we assume that $p(x)$ is further qualified to be a primitive polynomial such that α is a primitive element in GF(p^L). For simplicity, let $\mathbf{v}_i$, $0 \le i \le p^L - 2$, denote the representative (column) vector of α^i based on the polynomial representation. Then, the following theorem asserts that the matrix representation $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ consists of L representative vectors with consecutive subscripts.

Theorem 1. *For $0 \le i \le p^L - 2$, the matrix representation $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ can be written as follows:*

$$\mathbf{C}^i = \begin{bmatrix} \mathbf{v}_i & \mathbf{v}_{i+1} & \cdots & \mathbf{v}_{i+L-1} \end{bmatrix}. \tag{8}$$

As $\mathbf{C}^{p^L-1} = \mathbf{I}_L$, we omit the modulo-$(p^L - 1)$ expressions on the exponent of $\mathbf{C}$ and subscript of $\mathbf{v}$ throughout this paper for brevity.

Proof. First, the matrix $\mathbf{C}^i$ can be characterized by multiplication iterations based on (6) as follows. When $2 \le i \le L$,

$$\mathbf{C}^i = \left[\begin{array}{c|ccccc} & -a_0 & p_0^{(1)} & p_0^{(2)} & \cdots & p_0^{(i-1)} \\ & -a_1 & p_1^{(1)} & p_1^{(2)} & \cdots & p_1^{(i-1)} \\ \mathbf{U} & \vdots & \vdots & \vdots & \ddots & \vdots \\ & -a_{L-2} & p_{L-2}^{(1)} & p_{L-2}^{(2)} & \cdots & p_{L-2}^{(i-1)} \\ & -a_{L-1} & p_{L-1}^{(1)} & p_{L-1}^{(2)} & \cdots & p_{L-1}^{(i-1)} \end{array}\right], \tag{9}$$

where $L \times (L - i)$ matrix $\mathbf{U} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{L-i} \end{bmatrix}$. Further, when $L + 1 \le i \le p^L - 2$,

$$\mathbf{C}^i = \begin{bmatrix} p_0^{(i-L)} & p_0^{(i-L+1)} & \cdots & p_0^{(i-1)} \\ p_1^{(i-L)} & p_1^{(i-L+1)} & \cdots & p_1^{(i-1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{L-2}^{(i-L)} & p_{L-2}^{(i-L+1)} & \cdots & p_{L-2}^{(i-1)} \\ p_{L-1}^{(i-L)} & p_{L-1}^{(i-L+1)} & \cdots & p_{L-1}^{(i-1)} \end{bmatrix}. \tag{10}$$

The entries in (9) and (10) iteratively qualify

$$p_0^{(1)} = -a_0 a_{L-1},\ p_j^{(1)} = a_{j-1} - a_j a_{L-1}, 1 \le j \le L-1 \tag{11}$$

and

$$\begin{aligned} p_0^{(k)} &= -a_0 p_{L-1}^{(k-1)}, \\ p_j^{(k)} &= p_{j-1}^{(k-1)} - a_j p_{L-1}^{(k-1)}, 1 \le j \le L-1, 2 \le k \le i-1. \end{aligned} \tag{12}$$

When $i = 0$, it can be easily checked that each vector in $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{L-1}\}$ is a unit vector such that the only non-zero entry 1 of $\mathbf{v}_i$ locates at $(i+1)$th row. Therefore, $\mathbf{C}^0 = \mathbf{I}_L = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{L-1}]$, and (8) holds. When $i = 1$, consider $\mathbf{v}_L$ with $p(\mathbf{C}) = \mathbf{0}$, i.e.,

$$a_0 \mathbf{I}_L + a_1 \mathbf{C} + a_2 \mathbf{C}^2 + \ldots + a_{L-1} \mathbf{C}^{L-1} + \mathbf{C}^L = \mathbf{0}. \tag{13}$$

Obviously, $\mathbf{v}_L = \begin{bmatrix} -a_0 & -a_1 & \ldots & -a_{L-1} \end{bmatrix}^{\mathrm{T}}$ and (8) holds.

Assume when $i = m$, (8) holds, i.e., $\mathbf{C}^m = \begin{bmatrix} \mathbf{v}_m & \mathbf{v}_{m+1} & \cdots & \mathbf{v}_{m+L-1} \end{bmatrix}$. The Lth column vector $\begin{bmatrix} p_0^{(m-1)} & p_1^{(m-1)} & \ldots & p_{L-1}^{(m-1)} \end{bmatrix}^{\mathrm{T}}$ of $\mathbf{C}^m$ based on (9) corresponds to the representative vector of $\mathbf{C}^{m+L-1}$, that is, the matrix $\mathbf{C}^{m+L-1}$ is equal to

$$p_0^{(m-1)} \mathbf{I}_L + p_1^{(m-1)} \mathbf{C} + \ldots + p_{L-2}^{(m-1)} \mathbf{C}^{L-2} + p_{L-1}^{(m-1)} \mathbf{C}^{L-1} \tag{14}$$

It remains to prove, by induction, that $\mathbf{C}^{m+1} = \begin{bmatrix} \mathbf{v}_{m+1} & \mathbf{v}_{m+2} & \cdots & \mathbf{v}_{m+L} \end{bmatrix}$. As the column vectors indexed from 1th to $(L-1)$th of matrix $\mathbf{C}^{m+1}$ are exactly same as the ones indexed from 2th to Lth of $\mathbf{C}^m$, it suffices to show that the Lth column vector of $\mathbf{C}^{m+1}$ corresponds to $\mathbf{v}_{m+L}$. The following is based on (13) and (14):

$$\begin{aligned}\mathbf{C}^{m+L} &= p_0^{(m-1)}\mathbf{C} + p_1^{(m-1)}\mathbf{C}^2 + \ldots + p_{L-2}^{(m-1)}\mathbf{C}^{L-1} + p_{L-1}^{(m-1)}\mathbf{C}^L \\ &= p_0^{(m-1)}\mathbf{C} + p_1^{(m-1)}\mathbf{C}^2 + \ldots + p_{L-2}^{(m-1)}\mathbf{C}^{L-1} - p_{L-1}^{(m-1)}(a_0\mathbf{I}_L + \ldots + a_{L-1}\mathbf{C}^{L-1}) \\ &= -a_0 p_{L-1}^{(m-1)}\mathbf{I}_L + (p_0^{(m-1)} - a_1 p_{L-1}^{(m-1)})\mathbf{C} + \ldots + (p_{L-2}^{(m-1)} - a_{L-1}p_{L-1}^{(m-1)})\mathbf{C}^{L-1}.\end{aligned}$$

It can be easily checked that $p_0^{(m)}$ and $p_j^{(m)}$ with $1 \le j \le L-1$ in $\mathbf{C}^{m+1}$ calculated by (12) exactly consist of the representative vector of $\mathbf{C}^{m+L}$, i.e., $\mathbf{v}_{m+L}$. This completes the proof. □

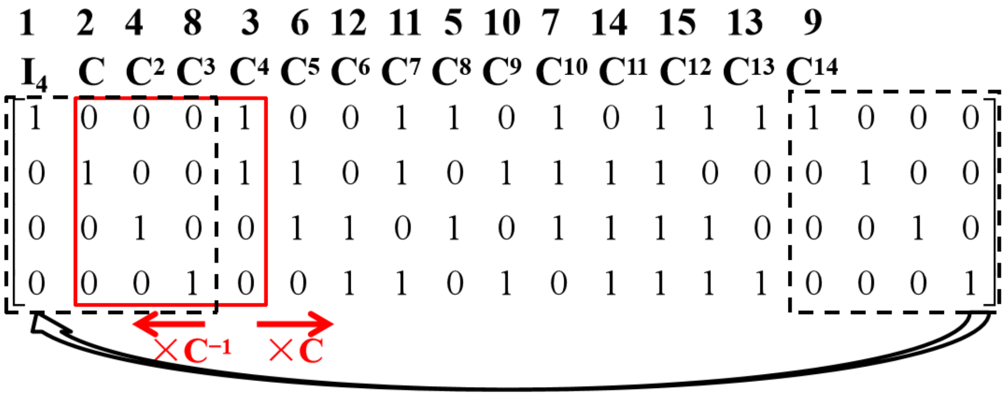
The above theorem draws an interesting conclusion that every non-zero matrix in $\mathcal{C}$ is composed of L representative vectors. Specifically, the first column vector of the matrix representation $\mathbf{C}^i$ is the representative vector of α^i, and its jth column vector, $1 \le j \le L$, corresponds to the representative vector of α^{i+j-1}. For the case $p = 2$, even though the above theorem is essentially same as Construction 4.1 and Lemma 4.2 in [13], its expression with the interplay of generator representation allows us to further devise a lookup table to pre-store the matrix representation with a smaller size.

In this table, we store p^L representative vectors with table size $L \times p^L$ and arrange them based on the power order of α with $0 \le i \le p^L - 2$. Note that the first column of matrix $\mathbf{C}^i$ can be indexed by vector $\mathbf{v}_i$ or $(i+1)$th column in this table, and the remaining columns of $\mathbf{C}^i$ can be obtained via subsequent $L-1$ column vectors based on Theorem 1. As a result, although this table only stores p^L vectors, it contains the whole matrix representations of GF(p^L) due to the inherent correlation among $\mathbf{C}^i$. The following Example 1 shows the explicit lookup table of GF(2^4) as an example.

Example 1. *Consider the field GF(2^4) and primitive polynomial $p(x) = 1 + x + x^4$ over GF(2). The companion matrix $\mathbf{C}$ is written as follows:*

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Then, the lookup table to store matrix representation $\mathbf{C}^i$ with $0 \le i \le 14$ is shown in Figure 1. In this figure, the solid "window" that currently represents the matrix $\mathbf{C}$ can be slid to the right or left to generate $\mathbf{C}^i$ with different i; meanwhile, the dashed box shows the cyclic property based on cyclic group $\{\mathbf{I}_4 = \mathbf{C}^{15}, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{14}\}$.



1	2	4	8	3	6	12	11	5	10	7	14	15	13	9				
$\mathbf{I}_4$	$\mathbf{C}$	$\mathbf{C}^2$	$\mathbf{C}^3$	$\mathbf{C}^4$	$\mathbf{C}^5$	$\mathbf{C}^6$	$\mathbf{C}^7$	$\mathbf{C}^8$	$\mathbf{C}^9$	$\mathbf{C}^{10}$	$\mathbf{C}^{11}$	$\mathbf{C}^{12}$	$\mathbf{C}^{13}$	$\mathbf{C}^{14}$				
1	0	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0
0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	0
0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0
0	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1

Figure 1. The lookup table to store the matrix representation $\mathbf{C}^i$ with $0 \le i \le 14$ for the field GF(2^4) and primitive polynomial $p(x) = 1 + x + x^4$.

Recall that in the lookup table of the matrix representation adopted in the latest release of the ISA-L library [12], the matrix representation of every element in GF(p^L) needs to be stored, so a total of L^2p^L p-ary elements need to be pre-stored. Compared with that, only an $L \times p^L$ p-ary matrix needs to be stored in the new lookup table, so the table size is reduced by a factor of $1/L$. Moreover, Theorem 1 implies the following useful corollaries of the matrix representation $\mathcal{C} = \{\mathbf{0}, \mathbf{C}^0, \mathbf{C}^1, \cdots, \mathbf{C}^{p^L-2}\}$ of GF(p^L).

Corollary 1. *Every vector in the vector space GF$(p)^L$ exactly occurs L times as a column vector in matrices of $\mathcal{C}$.*

Proof. As $\{\mathbf{I}_L, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{L-1}\}$ forms a polynomial basis of GF(p^L) over GF(p), the representative vectors of matrices in $\mathcal{C}$ are distinct. Consider a function $f : \{\mathbf{C}^i\} \rightarrow \{\mathbf{v}_i\}$. It can be easily checked that f is bijective, and $\mathbf{v}_i$ exactly corresponds to the jth column vector of $\mathbf{C}^{i-j+1}$ with $1 \leq j \leq L$. The zero vector of length L simply occurs L times in $L \times L$ matrix $\mathbf{0}$. □

Corollary 2. *For every GF(p^L), regardless of the choice of the primitive polynomial $p(x)$, the total number of zero entries in $\mathcal{C}$ remains unchanged as L^2p^{L-1}.*

The above two corollaries will be adopted to further demonstrate the advantages of binary matrix representation in vector LNC with $\mathcal{C}$.

4. Applications of Matrix Representation in Vector LNC

In this section, we focus on the study of vector LNC with binary matrices $\mathcal{C}$ and show the applications of matrix representation related to the aspects of random and deterministic coding.

4.1. Computational Complexity Comparison in Random LNC

Herein, the coding coefficients of random LNC are randomly selected from GF(2^L), which can provide a distributed and asymptotically optimal approach for information transmission, especially in unreliable or topologically unknown networks, such as wireless broadcast networks [17] or ad hoc network [18]. Recall that in polynomial and generator representations, the multiplication over GF(2^L) based on a lookup table requires two different types of calculation systems, so this table may not be utilized in resource-constrained edge devices. Therefore, under the same alphabet size 2^L, we first theoretically compare the random coding complexity between conventional LNC over $\{\beta = \sum_{l=0}^{L-1} v_l\alpha^l\}$ and vector LNC over $\mathcal{C}$ without lookup table, from the perspective of required binary operations.

To keep the same benchmark for complexity comparison, we adopt the following assumptions.

- We assume that an all-1 binary vector $\mathbf{m}$ as information will multiply $2^L - 1$ non-zero coding coefficients selected from $\{\beta = \sum_{l=0}^{L-1} v_l\alpha^l\}$ and $\mathcal{C}$, which can be simulated as encoding process. The complexity is the total number of binary operations that $2^L - 1$ multiplications take.
- We shall ignore the complexity of a shifting or permutation operation on the binary vector $\mathbf{m}$, which can be efficiently implemented.
- We only consider the standard implementation of multiplication in GF(2^L) by polynomial multiplication modulo and primitive polynomial $p(x) = a_0 + a_1x + \cdots + a_{L-1}x^{L-1} + x^L$ with η non-zero $a_i, 0 \leq i \leq L-1$, instead of considering other advanced techniques such as the FFT algorithm [19].

We first consider the encoding scheme with coefficients selected from $\{\beta = \sum_{l=0}^{L-1} v_l \alpha^l\}$. Assume that α is a root of $p(x)$ and every element β in GF(2^L) can be expressed as $g(\alpha)$, where $g(x)$ represents a polynomial over GF(2) with a degree less than L. An all-1 binary information vector $\mathbf{m}$ can be expressed as $\alpha^{L-1} + \alpha^{L-2} + \cdots + \alpha^2 + \alpha + 1$. We can divide the whole encoding process into two parts: multiplication and addition. In the multiplication part, the complexity of shifting operations is ignored, and one polynomial $\mathbf{m}\alpha^i$ in $\mathbf{m}g(\alpha)$ will modulo $p(x)$ i times and take $i\eta$ binary operations. Because every $\alpha^i, 1 \le i \le L-1$ occurs 2^{L-1} times among all $g(\alpha)$ in GF(2^L), it will take $\sum_{1 \le i \le L-1} i\eta \times 2^{L-1}$ binary operations to compute $\mathbf{m}\alpha^i$. In the addition part, it takes $(j-1)L$ binary operations to compute the additions between j binary vectors $\mathbf{m}\alpha^i$ with distinct i. Note that the number of distinct $g(\alpha)$ with j non-zero terms is $\binom{L}{j}$ in GF(2^L). Therefore, the traverse of $g(\alpha)$ will take an extra $\sum_{1 \le j \le L} \binom{L}{j} \times (j-1)L$ binary additions to compute $\mathbf{m}g(\alpha)$. In total, the complexity of this scheme is shown as follows:

$$\sum_{1 \le i \le L-1} i(\eta \times 2^{L-1} + L \times \binom{L}{i+1}). \tag{15}$$

Next, we consider the encoding scheme with coefficients selected from $\mathcal{C}$, whose complexity of encoding process depends on the total number of 1 in $\mathbf{C}^i$ with $0 \le i \le 2^L - 2$. In this framework, it is worthwhile to note that every $\mathbf{C}^i$ in $\mathcal{C}$ is full-rank and can extract a permutation matrix. Since the complexity of permutational operations is ignored, based on Proposition 2, the complexity of encoding process over $\mathcal{C}$ is shown as follows:

$$L^2 \times 2^{L-1} - L \times (2^L - 1) = 2^{L-1}(L^2 - 2L) + L. \tag{16}$$

For any primitive polynomial $p(x)$, $\eta \ge 2$. With $3 \le L \le 12$, Table 2 lists the *average* number of binary operations *per symbol* in two schemes. Specifically, every value calculated by Equation (15) and (16) has divided the alphabet size 2^L, and we can find that in random coding, the vector LNC via matrix representation can theoretically reduce at least half of the coding complexity to achieve multiplications under the same alphabet size 2^L.

Table 2. Average number of binary operations per symbol with parameter $\eta = 2$.

L	3	4	5	6	7	8	9	10	11	12
$\mathcal{C}$	1.88	4.25	7.66	12.09	17.55	24.03	31.52	40.01	49.51	60.01
$\sum_{l=0}^{L-1} v_l \alpha^l$	4.88	10.25	17.66	27.09	38.55	52.03	67.52	85.01	104.51	126.01
rate	38.5%	41.5%	43.3%	44.6%	45.5%	46.2%	46.7%	47.1%	47.3%	47.6%

4.2. The Special Choices of Binary $p(x)$ and Sparse $\mathbf{C}^i$

In addition to the random coding, a deterministic LNC where we pay a broader concern to reduce the computational complexity can also carefully design some special coding operations which can be efficiently implemented, such as circular shift [5,6] or permutation [7]. In this subsection, different from random choice of coefficients, we will carefully design the choices of binary primitive polynomial $p(x)$ and sparse matrices $\mathbf{C}^i$ in $\mathcal{C}$ based on the unveiled properties in Sec. III. We illustrate that the choice of $p(x)$ can influence the distributions of matrices $\mathbf{C}^i$ with different numbers of non-zero entries. Then, based on a proper $p(x)$, an algorithm is proposed to obtain a subset of $\mathcal{C}$, which contains a series of relatively sparse matrices in $\mathcal{C}$.

When $p = 2$, the entries in representative vectors based on Equation (11) and (12) will, respectively, degenerate as follows:

$$\begin{aligned} p_0^{(1)} &= a_{L-1}, \\ p_j^{(1)} &= a_{j-1} + a_j a_{L-1} = a_{j-1} + a_j p_0^{(1)}, 1 \leq j \leq L-1. \end{aligned} \tag{17}$$

and

$$\begin{aligned} p_0^{(k)} &= p_{L-1}^{(k-1)}, \\ p_j^{(k)} &= p_{j-1}^{(k-1)} + a_j p_{L-1}^{(k-1)} = p_{j-1}^{(k-1)} + a_j p_0^{(k)}, 1 \leq j \leq L-1, 2 \leq k \leq i-1. \end{aligned} \tag{18}$$

with a_0 must be 1 in $p(x)$. Based on the above two equations, consider two adjacent representative vectors $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$. When the last entry $p_{L-1}^{(k-1)}$ in $\mathbf{v}_{k-1}$ is equal to 0, the entries in $\mathbf{v}_k$ follow

$$p_0^{(k)} = 0,\ p_j^{(k)} = p_{j-1}^{(k-1)}, 1 \leq j \leq L-1,$$

which means that $\mathbf{v}_k$ can be generated by downward circular shift to $\mathbf{v}_{k-1}$. When the last entry $p_{L-1}^{(k-1)}$ equals 1, the entries in $\mathbf{v}_k$ follow

$$p_0^{(k)} = 1,\ p_j^{(k)} = p_{j-1}^{(k-1)} + a_j, 1 \leq j \leq L-1.$$

Therefore, the difference between $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$ in Hamming weight is no more than $\eta - 1$, where η represents the number of non-zero $a_i, 0 \leq i \leq L-1$ in primitive polynomial $p(x)$.

Note that for the matrix representation of every GF(2^L), the total number of 1 in $\mathcal{C}$ is always $L^2 \times 2^{L-1}$ regardless of the choice of binary $p(x)$. However, the value of η will influence the distributions of sparse matrices in $\mathcal{C}$. Based on (17) and (18), we can intuitively deduce that with smaller η, the sparse matrices in $\mathcal{C}$ will be more concentrated distribution. Since the identity matrix $\mathbf{I}_L$ with L non-zero entries is the sparsest full-rank matrix, we utilize Algorithm 1 to choose 2^{L-s} matrices in $\mathcal{C}$, which can be good candidates as coding coefficients of a practical coding scheme over GF(2^L).

Algorithm 1 The choice of sparse matrices over GF(2^L)

```
Initialize $S$ as an empty set of $L \times L$ binary matrix
    $S \leftarrow \mathbf{0}$
    $S \leftarrow \mathbf{I}_L$
    generate matrix $\mathbf{C}$ based on $p(x)$
    generate matrix $\mathbf{C}^{-1}$ based on Equation (18)
    define matrix $\hat{\mathbf{C}} = \mathbf{C}$
    define matrix $\hat{\mathbf{C}}^{-1} = \mathbf{C}^{-1}$
    define integer $s < L$: the required size of $\mathcal{C}_s$
for $i = 1 : 2^{L-s-1} - 1$
    $S \leftarrow \hat{\mathbf{C}}$
    $S \leftarrow \hat{\mathbf{C}}^{-1}$
    $\hat{\mathbf{C}} = \hat{\mathbf{C}} \times \mathbf{C}$
    $\hat{\mathbf{C}}^{-1} = \hat{\mathbf{C}}^{-1} \times \mathbf{C}^{-1}$
    $i = i + 1$
end
return $S$
```

In Algorithm 1, the multiplications using $\mathbf{C}$ or $\mathbf{C}^{-1}$ can be easily achieved by sliding the "window" right or left, respectively, as shown in Figure 1. Let $\mathcal{C}_s$ denote this subset of $\mathcal{C}$ and the 2^{L-s} matrices in $\mathcal{C}_s$ can be written as $\{\mathbf{0}, \mathbf{I}_L, \mathbf{C}^i, \mathbf{C}^{-i}\}$ with $1 \leq i \leq 2^{L-s-1} - 1$. Then, Table 3 lists the ratio of the total number of 1 between $\mathcal{C}_s$ and $\mathcal{C}$ with $s = 1, 2$. We can find that the 2^{L-s} matrices, which are special choices using Algorithm 1, indeed contain less 1 than the other matrices in $\mathcal{C}$.

Table 3. Ratio of total numbers of 1 between $\mathcal{C}_s$ and $\mathcal{C}$.

L	$p(x)$	η	$s = 1$	$s = 2$
3	$X^3 + X + 1$	2	0.3056	0.0833
4	$X^4 + X + 1$	2	0.3438	0.1094
5	$X^5 + X^2 + 1$	2	0.3800	0.1200
6	$X^6 + X + 1$	2	0.4410	0.1372
7	$X^7 + X + 1$	2	0.4585	0.1987
8	$X^8 + X^4 + X^3 + X^2 + 1$	4	0.4635	0.2235
9	$X^9 + X^4 + 1$	2	0.4777	0.2148
10	$X^{10} + X^3 + 1$	2	0.4950	0.2382
11	$X^{11} + X^2 + 1$	2	0.4898	0.2325
12	$X^{12} + X^6 + X^4 + X + 1$	4	0.4977	0.2421
13	$X^{13} + X^4 + X^3 + X + 1$	4	0.4906	0.2469
14	$X^{14} + X^5 + X^3 + X + 1$	4	0.4975	0.2500
15	$X^{15} + X + 1$	2	0.4979	0.2462
16	$X^{16} + X^5 + X^3 + X^2 + 1$	4	0.4978	0.2490

Moreover, in Figure 2, we numerically analyze the relationship between the number of 1 in each matrix and the corresponding number of matrices under the alphabet size 2^{12}. For all candidates of binary primitive polynomials, we choose four representative $p(x)$ with different $\eta = 4, 6, 8, 10$ and the specific polynomials are shown as follows:

$$\begin{aligned}
p_1(x) &= x^{12} + x^6 + x^4 + x^1 + 1 \\
p_2(x) &= x^{12} + x^7 + x^6 + x^5 + x^3 + x^1 + 1 \\
p_3(x) &= x^{12} + x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + x^1 + 1 \\
p_4(x) &= x^{12} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + x^1 + 1
\end{aligned}$$

As all the matrices in $\mathcal{C}$ are full-rank, the value range of the x-axis should be $[12, 132]$, and we restrict it to $[40, 100]$ to highlight the distributions. These four curves illustrate that with η increasing, the distribution variance of the number of 1 in a matrix will decrease, that is, the number of matrices with an average number of 1, i.e., 70–80, will increase and the number of relatively sparse or dense matrices will decrease. As a result, a smaller η of $p(x)$ not only infers a more concentrated distribution but also more amounts for sparse matrices in $\mathcal{C}$; then, we can select parameter s according to practical requirements and obtain $\mathcal{C}_s$ using Algorithm 1.

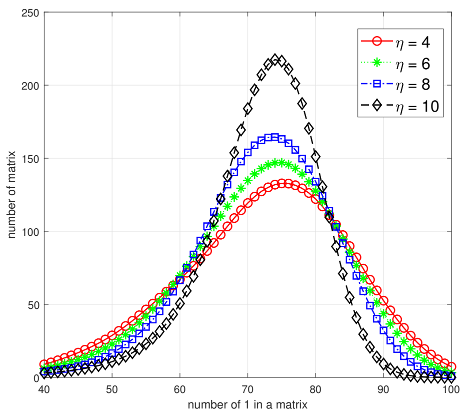


Figure 2. The distribution of sparse matrices in $\mathcal{C}$ with different $\eta = 4, 6, 8, 10$.

5. Conclusions

Compared with the classical result, the paraphrase of matrix representation in this paper focuses more on inherent correlation among matrices and a lookup table to pre-store the matrix representation with a smaller size is devised. This work also identifies that the total number of non-zero entries in $\mathcal{C}$ is a constant number, which motivates us to demonstrate the advantages of binary matrix representation in vector LNC. In the applications of matrix representation, we first theoretically demonstrate the vector LNC via matrix representation can reduce at least half of the coding complexity compared with conventional one over GF(2^L). Then, we illustrate the influence of η, i.e., the number of non-zero item in $p(x)$, on the amounts and distributions of sparse matrices in $\mathcal{C}$ and propose an algorithm to obtain sparse matrices which can be good candidates as coding coefficients of a practical vector LNC scheme.

Author Contributions: Methodology, H.T.; Software, S.J.; Writing—original draft, H.T.; Writing—review & editing, H.L.; Visualization, W.L.; Supervision, Q.S.; Funding acquisition, Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China under Grants U22A2005, 62101028 and 62271044, and by the Fundamental Research Funds for the Central Universities under Grant FRF-TP-22-041A1.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, S.-Y.R.; Yeung, R.W.; Cai, N. Linear network coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
2. Ebrahimi, J.B.; Fragouli, C. Algebraic algorithm for vecor network coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 996–1007. [CrossRef]
3. Sun, Q.T.; Yang, X.; Long, K.; Yin, X.; Li, Z. On vector linear solvability of multicast networks. *IEEE Trans. Commun.* **2016**, *64*, 5096–5107. [CrossRef]
4. Etzion, T.; Wachter-Zeh, A. Vector network coding based on subspace codes outperforms scalar linear network coding. *IEEE Trans. Inf. Theory* **2018**, *64*, 2460–2473. [CrossRef]
5. Tang, H.; Sun, Q.T.; Li, Z.; Yang, X.; Long, K. Circular-shift linear network coding. *IEEE Trans. Inf. Theory* **2019**, *65*, 65–80. [CrossRef]
6. Sun, Q.T.; Tang, H.; Li, Z.; Yang, X.; Long, K. Circular-shift linear network codes with arbitrary odd block lengths. *IEEE Trans. Commun.* **2019**, *67*, 2660–2672. [CrossRef]
7. Tang, H.; Zhai, Z.; Sun, Q.T.; Yang, X. The multicast solvability of permutation linear network coding. *IEEE Commun. Lett.* **2023**, *27*, 105–109. [CrossRef]
8. Wardlaw, W.P. Matrix representation of finite field. *Math. Mag.* **1994**, *67*, 289–293. [CrossRef]
9. Longhair: $O(N^2)$ Cauchy Reed-Solomon Block Erasure Code for Small Data. 2021. Available online: https://github.com/catid/longhair (accessed on 1 July 2024).
10. Plank, J.S.; Simmerman, S.; Schuman, C.D. Jerasure: A Library in c/c++ Facilitating Erasure Coding for Storage Applications, Version 1.2; Technical Report CS-08-627; University of Tennessee: Knoxville, TN, USA, 2008.
11. Luo, J.; Shrestha, M.; Xu, L.; Plank, J.S. Efficient encoding schedules for XOR-based erasure codes. *IEEE Trans. Comput.* **2014**, *63*, 2259–2272. [CrossRef]
12. Intel® Intelligent Storage Acceleration Library. 2024. Available online: https://github.com/intel/isa-l/tree/master/erasurecode (accessed on 1 June 2024).
13. Blomer, J.; Kalfane, M.; Karp, R.; Karpinski, M.; Luby, M.; Zuckerman, D. *An XOR-Based Erasure-Resilient Coding Scheme*; Technical Report TR-95-048; University of California at Berkeley: Berkeley, CA, USA, 1995.
14. Plank, J.S.; Xu, L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications. In Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 24–26 July 2006; pp. 173–180.
15. Zhou, T.; Tian, C. Fast erasure coding for data storage: A comprehensive study of the acceleration techniques. *ACM Trans. Storage (TOS)* **2020**, *16*, 1–24. [CrossRef]
16. Lidl, R.; Niederreiter, H. *Finite Fields*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1997.

17. Su, R.; Sun, Q.T.; Zhang, Z. Delay-complexity trade-off of random linear network coding in wireless broadcast. *IEEE Trans. Commun.* **2020**, *68*, 5606–5618. [CrossRef]
18. Asterjadhi, A.; Fasolo, E.; Rossi, M.; Widmer, J.; Zorzi, M. Toward network coding-based protocols for data broadcasting in wireless ad hoc networks. *IEEE Trans. Wirel. Commun.* **2010**, *9*, 662–673. [CrossRef]
19. Gao, S.; Mateer, T. Additive fast Fourier transforms over finite fields. *IEEE Trans. Inf. Theory* **2010**, *56*, 6265–6272. [CrossRef]