# Algebraic Algorithms for Vector Network Coding

Javad Ebrahimi B. and Christina Fragouli
School of Computer and Communication Sciences
EPFL, Lausanne, Switzerland.
Email: {javad.ebrahimi,christina.fragouli}@epfl.ch

***Abstract*—We develop new algebraic algorithms for scalar and vector network coding. In vector network coding, the source multicasts information by transmitting vectors of length $L$, while intermediate nodes process and combine their incoming packets by multiplying them with $L \times L$ coding matrices that play a similar role as coding coefficients in scalar coding. We start our work by extending the algebraic framework developed for multicasting over graphs in [1] to include operations over matrices; we build on this generalized framework, to provide a new approach for both scalar and vector code design which attempts to minimize the employed field size and employed vector length, while selecting the coding operations. Our algorithms also lead as a special case to network code designs that employ structured matrices.**

***Index Terms*—Algebraic framework, alphabet size, multivariate polynomials, network multicast, structured matrices, vector network coding.**

## I. Introduction

In the seminal paper [1], Köetter and Médard translated the problem of network code design to an algebraic problem. This algebraic framework provided the theoretical foundation for the development of most of the finite length coding results and practical algorithms for network coding today. In this paper, we show that this framework naturally extends to incorporate vector communication; building on this extended framework, we propose new algebraic code designs for scalar and vector network coding.

In scalar network coding, when multicasting to $N$ receivers at rate $h$, the source transmits $h$ scalar values over some finite field. The size of the employed finite field is a design parameter, that for complexity considerations we typically desire to minimize. Intermediate network nodes linearly combine their received symbols by multiplying them with scalar coefficients, called coding coefficients in the literature. The network code design consists of selecting the coding coefficients, and the finite field of operation, so that each receiver has a full rank set of equations to solve and can thus retrieve the source symbols [1].

In vector network coding, the source transmits $h$ vectors of length $L$, where the elements of the vectors are over a fixed finite field $\mathbb{F}_q$, for example, the binary field $\mathbb{F}_2$. Intermediate network nodes perform coding operations over vectors, namely, multiply their incoming vectors with $L \times L$ coding matrices and then add them to create the new vectors that they propagate towards the destinations. That is, intermediate nodes linearly combine their incoming vectors using coding matrices, where these matrices play a similar role as scalar coding coefficients in traditional algebraic network coding. The code design consists in selecting the length $L$ and the $L \times L$ coding matrices so that each receiver receives information at rate $h$.

This work was supported by the Swiss National Science Foundation Award No PP002110483 and the ERC Starting Grant Project NOWIRE ERC-2009-StG-240317.

Scalar operations over a field of size $q^L$ can be translated to vector operations employing $L \times L$ matrices using a well known mapping between finite fields operations and operations using matrices and vectors (see [21], chapter II, page 78). Thus, code designs for scalar network coding over a field $\mathbb{F}_{q^L}$ can be directly translated to code designs for vector network coding using $L \times L$ matrices with elements in $\mathbb{F}_q$. However, directly designing codes for vector network coding can still be useful; indeed, there exist $q^{L^2}$ $L \times L$ matrices over $\mathbb{F}_q$, while a translation from scalar coding only employs $q^L$ of these matrices. Thus, vector network coding offers a larger space of choices for optimizing cost parameters, such as the operational complexity, or the communication block length. Our work takes small steps in exploring this potential, using a subset of all possible matrices; we believe that the potential of vector coding is much beyond what this work achieves.

An alternative motivation to study vector network coding is that, in some networks, we may be restricted to employ operations over a fixed finite field $\mathbb{F}_q$; however, we may be allowed to vary the length of the vectors we process. One such case might be in practical networks, where intermediate nodes might need to be equipped in advance with the capability of operations over a fixed field. Scalar network coding requires operations over a finite field of size that grows with the number of receivers; to increase the size of the employed finite field, say from $\mathbb{F}_{q^L}$ to $\mathbb{F}_{q^{L+1}}$, we may need to communicate to all intermediate network nodes new multiplication and addition tables; on the contrary, with vector network coding, we would simply need to increase the size of the vectors we process from $L$ to $L+1$, while always performing operations over the same base field $\mathbb{F}_q$. Another such case may occur in deterministic networks, see [13], [14].

We start our work by extending the algebraic framework developed for multicasting over graphs in [1] to include operations over matrices. Building on this generalized framework, our contributions include:

- We provide a polynomial time algorithm for the design of coding matrices of vector network coding when multicasting to $N$ receivers. Our metric of optimization is the smallest size $L$ such that there exist $L \times L$ coding matrices that allow all $N$ destinations to successfully

decode the source information. The size of these matrices plays the same role as the size of the finite field in traditional network coding. Our algorithm reduces the problem of finding a small size $L$ to the problem of finding a small degree co-prime factor of an algebraic polynomial, and may lead to solutions not possible with using scalar network coding, as illustrated through examples in Sections V and VI. We note that the potential of vector coding was already noticed in several papers, see for example [5], [16], [17], [31], [35], and we discuss the differences with our work in Section II-B.
- From [1]–[4], [6], [27], it follows that if we translate scalar to vector code designs, use of vectors of length $L = \lceil \log_2(N+1) \rceil$ is always sufficient. In our work we provide probabilistic arguments indicating that a much smaller length of operation can be possible. For example, in a fraction $\frac{1023}{1024}$ of polynomials derived from transfer matrices, we will be able to find in polynomial time binary coding matrices of size at most $3 \times 3$ that lead to a valid code.
- Our approach gives a new algorithm for scalar network code design, that operates in polynomial time. This new algorithm jointly minimizes the employed field size while selecting the coding coefficients. In contrast, most[1] of the existing algorithms [1]–[4], [6], [27] first select a fixed finite field and then proceed to design the network codes over this predetermined field. Our approach builds on a connection between the problem of identifying the minimum field size required for network coding and the problem of finding the smallest co-prime factor of an algebraic polynomial.

The paper is organized as follows. Section II formalizes our problem statement and reviews related work. Section III generalizes the algebraic framework from scalar to vector coding. Section IV presents our proposed code designs for vector and scalar network coding, while Section V illustrates through examples the application of vector network coding and of our developed design tools. Section VI offers a comparison between scalar and vector network coding. Finally Section VII concludes the paper.

## II. Setup and Related Work

### A. Setup

In this paper, we consider the problem of designing scalar and vector network coding algorithms that allow us to multicast rate $h$ to $N$ receivers over directed acyclic graphs. We assume that the min-cut from a source $S$ to each receiver is at least $h$.

In general, network nodes can perform arbitrary operations; i.e., each network node can collect inputs, and send an arbitrary function of these inputs through each output. However, in this paper we will restrict our attention to linear operations, namely, each network code can perform linear combining with scalar coefficients for scalar network coding, and linear combining with matrix coefficients for vector network coding.

For simplicity, we present our designs for binary vector coding; the extension over arbitrary fields is straightforward, and we briefly discuss it in Section IV-A. Moreover, we will focus on acyclic networks, however, we note that the algebraic framework generalization that we will present in the next section applies for cyclic networks as well, using the same approach as in [1]. Finally, although in this paper we only consider the problem of multicasting, we believe that the tools we develop might be useful for other traffic scenaria as well.

### B. Related Work

Algebraic network coding for scalar coding was introduced in the seminal paper [1]. We here extend this framework to admit vector and scalar coding as special cases. This develops on the ideas we have presented in [13]–[15].

The potential of vector coding was already indicated through examples in several papers, see for example [5]. Designs specifically targeted to vector coding were, as far as we know, first proposed in [16], where vector coding was termed block network coding, and where the coding matrices were restricted to be binary interleaving matrices, leading to permute-and-add network codes. The code design in [16] is randomized over this specific set of structured matrices, and does not provide deterministic designs for finite length, in contrast to our approach. Very recently, the authors in [17] have proposed the use of rotation matrices, and also proposed designs specifically targeted to this set of matrices. Our work generalizes these results and provides a unifying framework which leads to additional structured network code designs.

Polynomial time designs for scalar network coding over graphs were proposed for example in [1]–[4]. An important difference between our work and these previous works is that we jointly minimize the employed field size, while selecting the coding coefficients, and may thus employ a much smaller field while still using a polynomial time algorithm. The only work that is close to ours in this sense is the work in [36]. The authors in [36] look at minimizing the field size that is employed by the LIF algorithm in [2], in contrast to our work that follows the algebraic approach. The algorithm in [36] starts from a binary field, and moves to an extension field when necessary and as the algorithm proceeds. Note that a field $2^{k_1}$ is an extension field of $2^{k_2}$ only if $k_2$ divides $k_1$. Thus, this algorithm only examines the fields of size 2, $2^2$, $2^4$, $2^8$, $2^{16}$, $2^{32}$, etc., and not all fields of size $2^3$, $2^5$, $2^6$, $2^7$, $2^9$, $2^{10}$, $2^{11}$,... etc. In contrast, our algorithms for scalar network coding examines all fields (even fields that have different characteristic). Additionally, in our work, we show a double exponential convergence assuming all polynomials occur with the same probability, while no such analysis is provided for the algorithm in [36].

## III. Algebraic Framework

We here first review the algebraic framework for scalar network coding in [1]; we then proceed to discuss how it can be extended to incorporate vector network coding.

[1]As far as we know, only the algorithm in [36] attempted to jointly minimize the field size and the code design before; we compare with this in Section II-B.

### *A. Review of the algebraic framework for scalar coding [1]*

In scalar coding, the source has $h$ symbols $\{u_1,\dots,u_h\}$ in a field $\mathbb{F}_q$ to multicast to $N$ receivers. Intermediate nodes linearly combine their received information using coding coefficients $\{X_k\}$ from the same field $\mathbb{F}_q$.

Consider a directed acyclic graph $G=(V,E)$, and each edge of the graph as a memory element that stores an intermediate information symbol. Assume that the source node has also $h$ auxiliary incoming edges, each bringing one of the $h$ symbols $\{u_1,\dots,u_h\}$. Then, if we consider a specific receiver $j$, our network acts as a linear system with $h$ inputs (the $h$ source symbols), $h$ outputs (that the receiver observes), and $\mu=|E|$ memory elements. This system is described by the following set of finite-dimensional state-space equations:

$$\begin{aligned}\mathbf{s}_{k+1}&=\mathbf{A}\mathbf{s}_k+\mathbf{B}\mathbf{u}_k\\ \mathbf{y}_k&=\mathbf{C}_j\mathbf{s}_k+\mathbf{D}_j\mathbf{u}_k,\end{aligned}\tag{1}$$

where $\mathbf{s}_k$ is the $\mu\times 1$ state vector at time $k$, $\mathbf{y}_k$ is the $h\times 1$ output vector, $\mathbf{u}_k$ is the $h\times 1$ input vector, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}_j$, and $\mathbf{D}_j$ are matrices with appropriate dimensions which we discuss in the following.

Matrix $\mathbf{A}$ of dimension $\mu\times\mu$ is common for all receivers and reflects the network topology, the way the edges (memory elements) are connected. Each entry $\mathbf{A}_{\ell j}$ of matrix $\mathbf{A}$ is either zero, one, or an unknown variable in the set $\{X_i\}$, the coefficient that will multiply the information symbol from edge $j$ as it flows through edge $\ell$. We will assume that we have in total $\nu$ such coefficients; clearly, $\nu\le\mu^2$. Matrix $\mathbf{B}$ is also common for all receivers and reflects the way the inputs (sources) are connected to our graph. Matrices $\mathbf{C}_j$ and $\mathbf{D}_j$ respectively express how the outputs that receiver $j$ observes, depend upon the state variables and the inputs. Matrices $\mathbf{B}$, $\mathbf{C}_j$, and $\mathbf{D}_j$ can be chosen to be binary matrices by possibly introducing auxiliary vertices and edges. Without loss of generality, we will assume that $\mathbf{D}_j=\mathbf{0}$ (this we can again do by possibly adding auxiliary edges and increasing the size $\mu$ of the state space).

A standard result in linear system theory gives us the transfer matrix $\mathbf{G}_j(\mathcal{D})$:

$$\mathbf{G}_j(\mathcal{D})=\mathbf{C}_j(\mathcal{D}^{-1}\mathbf{I}-\mathbf{A})^{-1}\mathbf{B},\tag{2}$$

where $\mathcal{D}$ is the indeterminate delay operator. In this paper, we will focus our attention to acyclic graphs, and thus we will assume unit delay, to obtain the $h\times h$ transfer matrix for receiver $j$:

$$\mathbf{M}_j=\mathbf{C}_j(\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}.\tag{3}$$

However, the generalization of the algebraic framework also holds for graphs with cycles.

By accordingly ordering the elements of the state space vector, matrix $\mathbf{A}$ becomes strictly upper triangular for acyclic graphs, and therefore, nilpotent, that is, $\mathbf{A}^n=0$ for some positive integer $n$. Let $\Lambda$ denote the length of the longest path between the source and a receiver. Then $\mathbf{A}^{\Lambda+1}=0$. In other words,

$$(\mathbf{I}-\mathbf{A})^{-1}=\mathbf{I}+\mathbf{A}+\mathbf{A}^2+\ldots+\mathbf{A}^\Lambda.\tag{4}$$

This equation immediately implies that the elements of the transfer matrices $\mathbf{M}_j$ are multivariate polynomials in the unknown variables $\{X_i\}$, of degree at most $\Lambda$ in each variable, and also of total degree at most $\Lambda$. By total degree we refer to the maximum sum of the degrees across the variables in each monomial; for example the polynomial $f(X_1,X_2)=X_1^2X_2+X_1+X_1X_2^4$ has three monomials, total degree 5, degree 2 in the variable $X_1$ and degree 4 in the variable $X_2$.

Network code design amounts to selecting values in $\mathbb{F}_q$ for the variable entries $\{X_i\}$ in $\mathbf{A}$, so that all matrices $\mathbf{M}_j$, for $j=1\dots N$, are simultaneously full rank. Equivalently, if the determinant of the matrix $\mathbf{M}_j$ is the multivariate polynomial $f_j(X_1,X_2,\dots,X_\nu)$, we want to select values for the variables $\{X_k\}$ in $\mathbb{F}_q$ so that all the polynomials $f_j$ evaluate to a nonzero value. Note that the degree of this polynomial in each variable equals at most $h\Lambda$, and the total degree as well equals $h\Lambda$.

The matrices $\mathbf{M}_j$, for $j=1\dots N$, are simultaneously full rank if and only if the matrix

$$\mathbf{M}\triangleq\mathbf{M}_1\cdot\mathbf{M}_2\cdots\;\ldots\;\cdots\mathbf{M}_N\tag{5}$$

is full rank. Thus, we can equivalently select values for the variables $\{X_i\}$ so that the polynomial

$$f(X_1,X_2,\dots,X_\nu)=\det\mathbf{M}\tag{6}$$

of total degree at most $Nh\Lambda$ evaluates to a nonzero value.

It can be shown in [3], [4] that $\mathbf{M}_j$ is full rank if and only if the matrix

$$\bar{\mathbf{M}}_j=\begin{bmatrix}\mathbf{C}_j & \mathbf{0}\\ \mathbf{I}-\mathbf{A} & \mathbf{B}\end{bmatrix}\tag{7}$$

is full rank. Let

$$\bar{\mathbf{M}}\triangleq\bar{\mathbf{M}}_1\cdot\bar{\mathbf{M}}_2\cdots\;\ldots\;\cdots\bar{\mathbf{M}}_N,\tag{8}$$

and for the polynomials corresponding to determinants, let

$$\bar{f}_j(X_1,\dots,X_\nu))=\det(\mathbf{M}_j),\tag{9}$$

and

$$\bar{f}(X_1,\dots,X_\nu)=\det(\bar{\mathbf{M}}).\tag{10}$$

The polynomial $\bar{f}$ has degree at most $N$ in each of the variables; however it may have a total degree as large as $N\mu$, where $\mu=|E|$ is the number of edges in the graph. The algebraic code design can equivalently be expressed as selecting values for the variables so that the polynomial $\bar{f}$ evaluates to a nonzero value. Three algebraic formulations for the multicasting problem are summarized in the following table. All these formulations are equivalent.

**Scalar Algebraic Formulations [1]:**

(1) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_i\}$ from the field $\mathbb{F}_q$ so that all matrices $\mathbf{M}_j$ become simultaneously full rank.

(2) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_i\}$ from the field $\mathbb{F}_q$ so that the polynomial $f(X_1,\dots,X_\nu)$ in (6) of degree at most $h\Lambda N$ in each variable and of total degree at most $h\Lambda N$, evaluates to a nonzero value.

(3) Select a finite field $\mathbb{F}_q$ and values for the variables $\{X_i\}$ from the field $\mathbb{F}_q$ so that the polynomial $\bar{f}(X_1, \ldots, X_\nu)$ in (10) of degree at most $N$ in each variable and of total degree at most $\mu N$, evaluates to a nonzero value.

From the sparse zero lemma (see Lemma 1 in [25], and Lemma 4 in [4]), we can assign to the variables $\{X_i\}$ values in a finite field $\mathbb{F}_q$ of size larger than $N$ so that all transfer matrices $\mathbf{M}_j$ are simultaneously invertible. Provided that $q > N$, we can find such values deterministically in polynomial time, for example using the methods [1]–[4], [6]. The algorithms we will develop in this paper differ from the existing algorithms in the literature in that they jointly optimize for the finite field of operation and the specific values for the coding parameters.

## *B. Extension of the algebraic framework to vector coding*

In vector coding, the source simultaneously conveys $h$ vectors of length $L$ to the destination, where $L$ is a design parameter. We will denote these vectors as $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$. These vectors take values over a predetermined field $\mathbb{F}_q$. For example, in most of this paper we will focus on binary vector coding, where $\mathbb{F}_q = \mathbb{F}_2$. We will also focus in the case where all the vectors transmitted in the network have equal length $L$. The intermediate network nodes collect vectors of length $L$, linearly process them by multiplying them with coding matrices with values in the field $\mathbb{F}_q$, and then further propagate them. We will denote the $L \times L$ coding matrices as $\{X_i\}$.

Exactly similar to before, we can associate a state variable with every edge of the network, where now each state variable is a vector of length $L$, and write the state-space equations for receiver $j$ as

$$\begin{aligned} \mathbf{s}_{k+1}^B &= \mathbf{A}^B \mathbf{s}_k^B + \mathbf{B}^B \mathbf{u}_k^B \\ \mathbf{y}_k^B &= \mathbf{C}_j^B \mathbf{s}_k^B + \mathbf{D}_j^B \mathbf{u}_k^B. \end{aligned} \tag{11}$$

If the network has $\mu = |E|$ edges, in the above equations, $\mathbf{u}_k^B$ is the $Lh \times 1$ input vector that contains the $h$ vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$, $\mathbf{s}_k^B$ is the $L\mu \times 1$ vector that contains the $\mu$ state vectors, and $\mathbf{y}_k^B$ is a $Lh \times 1$ output vector. Matrices $\mathbf{A}^B$, $\mathbf{B}^B$, $\mathbf{C}_j^B$, and $\mathbf{D}_j^B$ are now block matrices of appropriate dimension, that contain blocks of size $L \times L$. Without loss of generality, we can assume that $\mathbf{D}_j^B$ is the all zero matrix. Matrices $\mathbf{B}^B$ and $\mathbf{C}_j^B$ are fixed block matrices, that have as elements either the $L \times L$ identity matrix $\mathbf{I}$ or the $L \times L$ all zero matrix $\mathbf{0}$, at the same positions where matrices $\mathbf{B}$ and $\mathbf{C}_j$ in (1) had 1 and 0, respectively. Matrix $\mathbf{A}^B$ has also the same structure as matrix $\mathbf{A}$ in (1) with the difference that where matrix $\mathbf{A}$ had a variable entry $X_i$, now matrix $\mathbf{A}^B$ has at the same entry the $L \times L$ coding matrix $X_i$.

Exactly similar to before, the $hL \times hL$ transfer matrix for receiver $j$ can be calculated as

$$\mathbf{M}_j^B = \mathbf{C}_j^B (\mathbf{I} - \mathbf{A}^B)^{-1} \mathbf{B}^B. \tag{12}$$

and $\mathbf{M}_j^B$ is full rank if and only if the matrix

$$\bar{\mathbf{M}}_j^B = \begin{bmatrix} \mathbf{C}_j^B & \mathbf{0} \\ \mathbf{I} - \mathbf{A}^B & \mathbf{B}^B \end{bmatrix} \tag{13}$$

is full rank. Let also

$$\mathbf{M}^B \triangleq \mathbf{M}_1^B \cdot \mathbf{M}_2^B \cdot \ \ldots \ \cdot \mathbf{M}_N^B \tag{14}$$

and

$$\bar{\mathbf{M}}^B \triangleq \bar{\mathbf{M}}_1^B \cdot \bar{\mathbf{M}}_2^B \cdot \ \ldots \ \cdots \bar{\mathbf{M}}_N^B. \tag{15}$$

We observe that the dimensions of matrices $\mathbf{M}_j^B$ and $\bar{\mathbf{M}}_j^B$ depend upon the size parameter $L$. The multicasting code design problem is to select the size parameter $L$ and the $L \times L$ coding matrices $\{X_i\}$ so that all matrices $\mathbf{M}_j^B$ for $j = 1 \ldots N$ are simultaneously full rank. We will denote the set of $L \times L$ matrices with elements over a field $\mathbb{F}_q$ as $M_L(\mathbb{F}_q)$. Thus, the matrices $\{X_i\}$ take values in $M_L(\mathbb{F}_q)$.

Since the matrices we consider have the same structure for the scalar and vector case, we will omit the superscript $B$, and refer for example to a matrix $\mathbf{A}$ that has as elements variables $\{X_i\}$ that take values in an algebraic structure, either a finite field $\mathbb{F}_q$ or $M_L(\mathbb{F}_q)$.

**Example III.1.** *For the butterfly network [7], [8], the transfer matrices to the two receivers are*

$$\mathbf{M}_1 = \begin{bmatrix} X_1 & X_2 \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} X_1 & X_2 \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

*In vector coding the transfer matrices are block matrices, and each element is an $L \times L$ matrix, where $\mathbf{0}$ is the all-zero and $\mathbf{I}$ the identity matrix. For the butterfly network it is sufficient to use $L = 1$.*

The following theorem helps relate the problem of vector code design to the problem of a polynomial evaluation. A proof of this theorem can be found in [22].

**Theorem III.2.** *Let $\mathbf{M}$ be an $hL \times hL$ matrix over a field. Suppose $\mathbf{M}$ is subdivided into $h^2$ blocks $\mathbf{M}_{i,j}, 1 \le i,j \le h$ each of which is an $L \times L$ matrix. Moreover, suppose that for all numbers $1 \le i, i', j, j' \le n$ we have $\mathbf{M}_{i,j} \cdot \mathbf{M}_{i',j'} = \mathbf{M}_{i',j'} \cdot \mathbf{M}_{i,j}$. Then det$(\mathbf{M})$ = det$(f(\mathbf{M}_{1,1}, \mathbf{M}_{1,2}, \ldots, \mathbf{M}_{n,n}))$ where $f(x_{1,1}, x_{1,2}, \ldots x_{n,n})$ = det$([x_{i,j}])$.*

Thus, if the matrices we chose for the variables $\{X_i\}$ are pairwise commuting, then, from Theorem III.2, $\det(\mathbf{M}_j) = \det(f_j(X_1, \ldots, X_\nu))$, $\det(\mathbf{M}) = \det(f(X_1, \ldots, X_\nu))$ and $\det(\bar{\mathbf{M}}_j) = \det(\bar{f}_j(X_1, \ldots, X_\nu))$. We will call, in this case, the polynomial

$$f_j(X_1, \ldots, X_\nu) : M_L(\mathbb{F}_2) \times \ldots \times M_L(\mathbb{F}_2) \to M_L(\mathbb{F}_2)$$

a *matrix* polynomial, to indicate that its evaluation results in an $L \times L$ matrix (and similarly for $f$ and $\bar{f}$). The vector code design problem can be cast as selecting the length $L$ and the commutative $L \times L$ matrices $\{X_i\}$ so that the matrix polynomials $f(X_1, \ldots, X_\nu)$ and $\bar{f}(X_1, \ldots, X_\nu)$ evaluate to an invertible matrix, as summarized in the following table.

**Vector Algebraic Formulations:**
(1) Select length $L$ and $L \times L$ matrices $\{X_i\}$ in $M_L(\mathbb{F}_q)$ so that all matrices $\mathbf{M}_j$ become simultaneously full rank.
(2) Select length $L$ and $L \times L$ commutative matrices $\{X_i\}$ in $M_L(\mathbb{F}_q)$ so that the matrix polynomial $f(X_1, \ldots, X_\nu)$

of degree at most $h\Lambda N$ in each variable and of total degree at most $h\Lambda N$, is an invertible matrix.
(3) Select length $L$ and $L \times L$ commutative matrices $\{X_i\}$ in $M_L(\mathbb{F}_q)$ so that the the matrix polynomial $\bar{f}(X_1, \ldots, X_\nu)$ of degree at most $N$ in each variable and of total degree at most $mN$, is an invertible matrix.

Note that although formulations (2) and (3) are equivalent, they lead only to a subset of the possible solutions, since they require the use of commutative matrices. Formulation (1) does not impose this assumption and can lead to solutions not possible with (2) and (3).

One approach for vector network coding is to find a solution for scalar coding over a field of size $q^L$, and then use a standard translation between scalar operations over a field of size $q^L$ to vector operations employing coding matrices in $M_L(\mathbb{F}_q)$ (see [34], chapter 7, page 424). For completeness, we include the formal proof of this mapping in Appendix A. This solution however, only uses $q^L$ of the matrices in $M_L(\mathbb{F}_q)$ and is more restrictive that the above formulations (1), (2) and (3); indeed, formulations (2) and (3) only restrict the matrices $X_i$ to be commutative, while there exist collections[2] of commutative matrices with more than $q^L$ members.

## IV. Code Design

In this section we develop our algebraic algorithms for vector and scalar network coding. As we mentioned before, we consider for the rest of this paper acyclic networks. For simplicity, we restrict our attention to binary vector network coding, where the vectors and the matrices have elements in the binary field, but the extension over arbitrary fields is straightforward, and we briefly discuss it at the end of Section IV-A.

Both for vector and scalar network coding, we start from the algebraic formulation described in Section III. That is, we construct the transfer matrices $\mathbf{M}_j$, $1 \le j \le N$, $\bar{\mathbf{M}}_j$, $1 \le j \le N$, and $\mathbf{M}$. Our algorithms for scalar network coding seek to assign scalar values to the variables $\{X_i\}$, over a field of size $q$ as small as possible. Similarly, for vector network coding, they seek to assign to these variables $L \times L$ binary coding matrices in $M_L(\mathbb{F}_2)$, of a size $L$ as small as possible.

The code design consists of two basic steps:

- *Step 1:* we reduce the multivariate polynomial $f(X_1, \ldots, X_\nu)$ to a single-variable polynomial $f(X)$, by expressing each variable $X_i$ as a polynomial $p_i(X)$ of the same variable $X$. We carefully select these polynomials so that the resulting polynomial $f(X)$ does not become identically zero;
- *Step 2:* for scalar network coding we select a scalar value for the variable $X$ from a finite field of size as small as possible, and for vector network coding we select a matrix for the variable $X$ of size as small as possible, so that the polynomials evaluate to a nonzero value for scalar coding, and to an invertible matrix for vector coding.

In the following, we first describe the code design for vector network coding, we then proceed to the design for scalar network coding, and finally explore the connections between the two algorithms.

### A. Code design for vector coding

*1) Algorithm Description:* We start by describing our algorithm, and then analyze its performance. The complexity calculation is provided by Lemma IV.6.

*Step 1: Assignment of polynomials to $\{X_i\}$*

1) Assume that the variables $\{X_i\}$ take scalar values. Using the matrix completion methods in [3], we can find an assignment of values to the variables $\{X_i = \alpha_i\}$, with $\{\alpha_i\}$ in a finite field $\mathbb{F}_q$ of size $q > 2^{\lceil \log_2 N \rceil}$, so that all matrices $\bar{\mathbf{M}}_j$ become invertible, i.e., $\det(\bar{\mathbf{M}}_j) \neq 0$, $j = 1 \ldots N$. Note that this assignment of values $\{X_i = \alpha_i\}$ also makes $\det(\mathbf{M}_j) \neq 0$, for $j = 1 \ldots N$, and $\det(\mathbf{M}) \neq 0$. That is,
$$f(X_1 = \alpha_1, \ldots, X_\nu = \alpha_\nu) \neq 0. \tag{16}$$
2) Assume that the field $\mathbb{F}_q$, where the values $\{\alpha_i\}$ belong, has size $q = 2^k$ with $k = \lceil \log_2 N \rceil + 1$. Using a standard representation of extension fields ( [21], chapter 1, page 2), we can express each value $\alpha_i \in \mathbb{F}_{2^k}$, identified in the previous step, as a binary polynomial $p_i(X)$ of degree at most $k-1$ in an indeterminate $X$. For example, we could have that $\alpha_1$ is expressed as $p_1(X) = X^2 + 1$, and $\alpha_\nu$ is expressed as $p_\nu(X) = X^3$. We substitute these polynomials in place of the variables $\{X_i\}$ in the transfer matrices $\mathbf{M}_j$ and the matrix $\mathbf{M}$.
3) We calculate the determinant of the matrix $\mathbf{M}$. Note that the entries of $\mathbf{M}$ are polynomials in a single variable $X$, and thus the determinant can be calculated efficiently, for example through interpolation as discussed in Lemma VII.3 in the Appendix. We then get a single variable polynomial $f(X)$, that equals
$$f(X) \triangleq f(X_1 = p_1(X), \ldots, X_\nu = p_\nu(X)). \tag{17}$$
For example,
$$f(X) = f(X_1 = X^2 + 1, \ldots, X_\nu = X^3).$$

It follows from (16) that the polynomial $f(X)$ in (17) is not identically zero. Moreover, from Section III, it has degree at most $N(k-1)h\Lambda$ in the variable $X$, where $\Lambda$ is the longest path length from the source to a receiver.

We underline that in the above procedure we never explicitly calculate the polynomial $f(X_1, \ldots, X_\nu)$, as this calculation is not polynomial time; instead, we directly calculate the polynomial $f(X)$ in (17).

Now consider the variables $\{X_i\}$ as $L \times L$ matrices, and assume we express each such matrix as the polynomial $p_i(X)$ we have previously identified, of an $L \times L$ matrix $X$. This

[2]For example, consider the set of all matrices of size $2k \times 2k$ with the properties that (1) every non-zero entry is located either on the main diagonal or on the $k \times k$ upper right block and (2) all the elements on the main diagonal are the same; these matrices are commutative, and their set has size $q^{k^2+1}$ which is larger than $q^{2k}$.

assignment ensures that the resulting matrix polynomial $f(X)$ in (17) is not identically zero. Our code design problem is now reduced to selecting the size parameter $L$ and a single matrix $X = \mathbf{A}$ so that the matrix $f(\mathbf{A})$ is invertible.

*Step 2: Assignment of value to X*

1) Find a polynomial $g(X)$ that is co-prime with $f(X)$, of degree $m$ as small as possible. We will prove in the analysis of our algorithm (section IV-A2) that we can always find such a $g(X)$ of degree $m \leq \log_2(N+1)$ in polynomial time.
2) If $g(X)$ has degree $m$, create an $m \times m$ matrix $\mathbf{A}$ so that $g(\mathbf{A}) = 0$, using for example the well known construction in Lemma IV.2.
3) Select $L = m$ and $X = \mathbf{A}$. The following Lemma IV.1 proves that for this selection, $f(\mathbf{A})$ is an invertible $m \times m$ matrix. Thus, each coding matrix $X_i$ is assigned the $L \times L$ matrix $p_i(\mathbf{A})$.

In the following, we denote by $\mathbb{F}_q[x]$ the ring of all polynomials in variable $x$ and with coefficients over a field $\mathbb{F}_q$.

**Lemma IV.1.** *Let $f(x)$, $g(x)$ be two relatively co-prime polynomials in $\mathbb{F}_q[x]$. If $\mathbf{A}$ is a matrix in $M_L(\mathbb{F}_q)$ and $g(\mathbf{A}) = 0$, then $f(\mathbf{A})$ is an invertible matrix.*

*Proof:* Since $\gcd(f(x), g(x)) = 1$, there exist polynomials $h_1(x)$, $h_2(x)$ so that $f(x)h_1(x) + g(x)h_2(x) = 1$. If we set $x = \mathbf{A}$ we get $f(\mathbf{A})h_1(\mathbf{A}) + g(\mathbf{A})h_2(\mathbf{A}) = \mathbf{I}$. Since $g(\mathbf{A}) = 0$, $f(\mathbf{A})h_1(\mathbf{A}) = \mathbf{I}$. ■

**Lemma IV.2.** *( [34], chapter 7, page 425) The $m \times m$ matrix*

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & 0 & \dots & 0 & -a_2 \\ 0 & 0 & 1 & \dots & 0 & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -a_{m-1} \end{bmatrix} \tag{18}$$

*has the characteristic polynomial $g(x) = x^m + a_{m-1}x^{m-1} + \ldots + a_0$ and thus satisfies $g(\mathbf{A}) = 0$.*

*2) Algorithm Analysis:* We now analyze the performance of our algorithm. Lemma IV.6 provides the complexity calculation of our algorithm, and shows that the required number of operations is polynomial in the number of variables $\nu$, the min-cut $h$, the number of receivers $N$ and the longest path length $\Lambda$ (in the following, when we refer to polynomial time, we mean that the complexity is polynomial in these network parameters).

Our algorithm also attempts to minimize the size $L$ of the employed coding matrices, which is equal to the degree $m$ of the lowest degree polynomial $g(X)$ co-prime to $f(X)$ we can find. In Theorem IV.3 and Lemma IV.4 we provide upper bounds on the degree $m$ of $g(X)$ that could be required (hard guarantees). In Lemma IV.5 we show that the fraction of polynomials of a given degree $n$ that have a co-prime polynomial of degree at most $m$, converges doubly exponentially (with $m$) to one. This strongly indicates that our algorithm will in the majority of cases result in a size much smaller than the upper bound in Theorem IV.3.

**Theorem IV.3.** *If $f(x)$ is a nonzero binary polynomial of degree $n$, then there exists an irreducible co-prime polynomial $g(x)$ of degree at most $\log_2(n+1) - 1$, and we can identify $g(x)$ in polynomial time in $n$. There also exist polynomials that require $g(x)$ to have this degree.*

*Proof:* As candidates for the polynomials $g(x)$, we are going to consider irreducible polynomials. The main observation is that, since $f(x)$ has a finite degree, it cannot have as factors an arbitrary number of irreducible polynomials. In particular, let $g_1, g_2, \ldots g_K$ be all the irreducible binary polynomials of degree at most $m$ then $\Pi_{j+1}^{K} g_j(x)$ divides $f(x)$, otherwise at least one of the $g_i$'s is co-prime with $f$.

In Appendix A, Lemmas VII.1 and VII.2, we prove that the summation of the degrees of all the irreducible binary polynomials of degree at most $m$ is $(1-2\epsilon)2^{m+1}$ for some small $\epsilon$. Then $f(x)$ must have degree larger than this summation, i.e., $2^{m+1} \leq n$, and the result follows.

It is also easy to see that we can find such a co-prime $g(x)$ in polynomial time. Indeed, the total number of such polynomials is at most $n+1$, and thus exhaustive search would suffice.

Finally, assume that $f(x)$ is indeed the product of all irreducible binary polynomials of degree at most $m$ - then $g(x)$ would need to have degree $m+1$ and the last statement follows. ■

*Observation:* in this theorem, we restricted our attention to irreducible polynomials $g(x)$. However, we could simply consider, as candidates for co-prime factors, all polynomials of degree $m$, for any $m$ such that $m(2^{m+1} - 1) \geq n$ (for example, $m = \log_2 n$), without verifying if a polynomial is irreducible,which can lead to a faster search. To design vector network codes we can directly use any polynomial, irrespective of whether it is irreducible or not; for the design of scalar code, that we will discuss in Section IV-B, we need to use an irreducible polynomial, but we can always factor the identified polynomial, and use one of the irreducible factors.

Theorem IV.3 implies that our algorithms always allows to operate using length $L$ upper bounded by $\log_2(n) \leq \log_2(Nh\Lambda\log_2(N))$. The following Lemma argues that our specific algorithm will in fact find such a co-prime factor of degree at most $\log_2(N+1)$. Although this is an improved bound for our algorithm, we believe the looser alternative upper bound in Theorem IV.3 might still be interesting, as it is independent of the employed technique to identify the polynomials $p_i(X)$.

**Lemma IV.4.** *The algorithm described in Section IV-A1 terminates by finding a co-prime factor of degree at most $L \approxeq \log_2(N+1)$.*

*Proof:* This can be trivially shown by applying results in [2], [4]. Namely, there exists an irreducible polynomial $h(X)$ that generates the field of size $q > 2^{\lceil \log_2 N \rceil}$ over which we make the scalar assignment in Step 1 of our algorithm. This polynomial has degree approximately $\log_2(N+1)$, and, since the scalar assignment results in a nonzero value, it is co-prime with the polynomial $f(X)$. ■

In Theorem IV.3, we proved that given a polynomial $f(x)$ of degree $n$, we can always find a co-prime polynomial $g(x)$ of degree $m = O(\log_2 n)$. We next show that, although $m = O(\log_2 n)$ is always sufficient, we can in many cases find a co-prime polynomial of much smaller degree than $O(\log_2 n)$.

**Lemma IV.5.** *The fraction of polynomials of degree $n$ for which there exists a co-prime polynomial of degree at most $m$, with $n > m$, converges doubly exponentially (with $m$) to one. In particular, this fraction is at least as large as*

$$1 - \prod_{i=1}^{m} \frac{1}{2^{i\zeta(i)}},$$

*where $\zeta(i)$ is the number of irreducible binary polynomials of degree $i$ and can be approximated by $\frac{2^i}{i}$.*

*Proof:* For a fixed polynomial $g(x)$ not identically zero $g(x)$ is a factor of $f(x)$ for a fraction of $\frac{1}{2^m}$ of all polynomials $f(x)$ of degree $n$, with $n > m$. This follows by observing that the remainder after dividing $f(x)$ with $g(x)$, can be any of the $2^m$ binary polynomials of degree smaller or equal to $m-1$. Moreover, we can divide the polynomials of degree $n$ to $2^m$ mutually exclusive and equally-sized sets, one corresponding to each possible remainder.

Let $g_1(x),\ g_2(x),\ \ldots,\ g_k(x)$ be pairwise co-prime polynomials. Therefore $f(x)$ is divisible by all of them if and only if it is divisible by their product. The fraction of non-zero polynomials $f$ that have all of the $g_i$'s as a factor is at most $\prod_{i=1}^{k} \frac{1}{2^{m_i}}$, where $m_i$ is the degree of $g_i(x)$. ■

For example, if we take $g_1(x) = x$ and $g_2(x) = x+1$, then $\frac{3}{4}$ of all polynomials will be co-prime with either $g_1$ and/or $g_2$. For the case of $g_1(x) = x$, $g_2(x) = x+1$, $g_3(x) = x^2+x+1$, the fraction increases to $\frac{15}{16}$, and if we consider all the irreducible polynomials of degree at most 3, the fraction becomes $\frac{1023}{1024}$. That is, if we consider the set of all polynomials of a given degree $n \geq 3$, a fraction of $\frac{1023}{1024}$ of these polynomials have a co-prime polynomial of degree $m \leq 3$.

As a result, if all polynomials of degree $n$ resulted from transfer matrices of networks with equal probability, in $\frac{1023}{1024}$ of such networks a binary matrix of size at most $3\times 3$ would lead to valid code. We note however that we currently do not know the probability distribution of polynomials that result from transfer matrices. Thus the above results might be pessimistic or optimistic.

**Lemma IV.6.** *The complexity of the algorithm is*

$$O(N(\nu+h)^3 \log_2(\nu+h) + \nu(\nu+h)^2 + (N \log_2 Nh\Lambda)^3).$$

*Proof:* Our algorithm consists of the following steps:

- For identifying the values $\{\alpha_i\}$, the matrix completion algorithm requires $O(N((\nu+h)^3 \log(\nu+h) + \nu(\nu+h)^2))$, where $\nu$ is the number of variables [3].
- To calculate the polynomial $f(x)$ using the determinant calculation in Lemma VII.3, we evaluate $x$ at different points. For each evaluation point $\beta_i$ we can compute the value $f(\beta_i)$ as the determinant[3] of the matrix $\mathbf{M}(\beta_i)$ in $O(h^3)$. So, in total we will have $O(h^3 N \log Nh\Lambda)$ operations for the determinant and we need $O((N \log Nh\Lambda)^3)$ operations to find the coefficients $\{c_i\}$ of the polynomial $f(x)$ (see also Lemma VII.3.)
- For a polynomial $f$ of degree $n = Nh\Lambda \log N$, to find a co-prime factor of degree at most $\log_2 n$, we need to perform approximately $n$ polynomial divisions, where each division can be achieved using complexity $n \log_2 n$.

■

*3) Vector coding over a field of size $\mathbb{F}_q$, with $q > 2$:* When the ground field is $\mathbb{F}_q$ for $q > 2$ the same algorithm and analysis can be applied. The only important difference is that the number of irreducible polynomials of degree $m$ and coefficients in $\mathbb{F}_q$ can be approximated by $\frac{1}{m}q^m$. As a result, the value $L$ we need to employ reduces as $q$ increases.

## *B. Code design for scalar coding*

*Step 1: Assignment of polynomials to $\{X_i\}$*

In the same way as in Step 1 in Section IV-A, we create the not-identically zero polynomial $f(X)$. We thus reduce the code design problem to the problem of finding a value $X = \alpha$ so that $f(\alpha) \neq 0$.

*Step 2: Assignment of value to $X$*

1) Similar to Step 2 in Section IV-A, we find an irreducible polynomial $g(X)$ that is co-prime with $f(X)$ of degree at most $m = \log_2 n$. In this case, however, it is important that the polynomial $g(x)$ is irreducible.
2) We consider the finite field of size $\mathbb{F}_{2^m}$ generated by the polynomial $g(X)$. We make the assignment

   $$X_i = p_i(X) \mod g(X).$$

   Thus, each $X_i$ is assigned a value in the field $\mathbb{F}_{2^m}$. The polynomial $f(X)$ evaluates to the nonzero value $f(X) \mod g(X)$.

That is, we assign to $X$ the value $\alpha$ in the finite field generated by $g(X)$, corresponding to the indeterminate $X$. The second step is what distinguishes our algorithms from the algebraic code designs in the literature. The solution we find from Step 1 may use an unnecessarily large field; step 2 aims to reduce the employed field size.

*Analysis*

The analysis is the same as in Section IV-A.

*Novelty of the algorithm*

As also mentioned in Section II-B, there exist several polynomial time designs for scalar network coding over graphs, see for example [1]–[4]. An important difference between our work and these previous works is the following. Elements of finite fields can be regarded as polynomials over a base field,

[3]These calculations assume operations over a fixed field size $q$, that remains constant; if not, we need to include a factor $\log_2 q$ to account for the field operations.

modulo an irreducible polynomial $g(x)$. When, in network code design, we make an assignment to the variables of a multivariate polynomial using elements from a finite field, we are replacing each variable with a polynomial of $x$ in a way that ensures the final result is not zero modulo $g(x)$. Notice that the final result, after the substitution, might be nonzero, and might only become zero once we apply the modulo $g(x)$ operation. In contrast, in our approach we do not pre-determine $g(x)$. We first ensure that the multivariate polynomial (derived from the transfer matrix) is nonzero. We then choose an $g(x)$ that it is co-prime with the resulting polynomial. We illustrate this we a simple example.

**Example IV.7.** *Consider a combination network with three receivers and transfer matrices of the form:*

$$T_1 = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \end{bmatrix}, \quad T_2 = \begin{bmatrix} X_1 & Y_1 \\ X_3 & Y_3 \end{bmatrix}, \quad T_3 = \begin{bmatrix} X_3 & Y_3 \\ X_2 & Y_2 \end{bmatrix}.$$

*Since we have three receivers, Step 1 that does not optimize the field size will search for a solution over the field $\mathbb{F}_p$ with $p > 3$. For example, using the field $\mathbb{F}_4$ we can get the following solution:*

$$X_1 = Y_2 = X, Y_1 = X_3 = 1, X_2 = Y_3 = 0.$$

*where $X$ is an element of $\mathbb{F}_4$ other than $0$ or $1$. Step 2 in our algorithm takes this scalar code and computes the product of the determinants of the transfer matrices; in our example the product is $X^2$. Then it finds a polynomial which is co-prime with $X^2$, for example $X+1$ and reduces all variables modulo this polynomial. Thus we get a solution over the binary field $\mathbb{F}_2$.*

Note that our algorithm can also be applied as an additional step to any other network code design algorithm, to attempt to reduce the employed field size.

### *C. Alphabet Size in Network Coding*

It is interesting to note that our algorithm reduces the problem of minimizing the alphabet size (finite field of operation) in scalar network coding, to the problem of finding a reduction of the polynomial $f(X_1, \ldots, X_\nu)$ to a single variable polynomial $f(X)$ that has a co-prime factor of degree as small as possible.

It was shown in [19], [20] that the problem of finding the minimum alphabet size is NP-hard, through a reduction from the graph coloring problem. The hardness in our equivalent algebraic formulation is expressed through the manner the reduction to a single variable polynomial is performed. This reduction can be performed in multiple ways, and what is the optimal way is not clear. For example, a polynomial $f_1(X)$ can have larger degree than a polynomial $f_2(X)$, however, $f_1(X)$ may have a smaller degree co-prime factor than $f_2(X)$, leading to the use of a smaller alphabet size.

## V. Application to Structured Matrices

In this section, we show how we can apply the vector network coding framework to use, as coding matrices, specific types of structured matrices that lead to low complexity implementations. Use of structured matrices has already been proposed in the literature, for permutation matrices in [16], and very recently for rotation matrices in [17]. Our designs give a different viewpoint in the utilization of such matrices, and allow to develop alternative designs.

The main idea that we apply in the following is that structured matrices have associated characteristic polynomials that are also structured; we can find a solution using our algorithms, provided at least one such structured polynomial is co-prime with the polynomials we can derive from transfer matrices.

### *A. Rotation Matrices over $\mathbb{F}_q$*

We are here interested in using rotation matrices as coding matrices, in order to achieve low complexity encoding at the network nodes. Use of rotation matrices over the binary field was originally proposed in [17]. We here consider a more general form of rotation matrices, over an arbitrary field, and show how our developed framework can be used in order to try to design network codes that employ such matrices.

**Definition 1.** *An $L \times L$ matrix $\mathbf{A}$ over a field $\mathbb{F}_q$ is called a t-th order rotation matrix with respect to the L-tuple $(c_1, c_2, \ldots, c_L)$ in $\mathbb{F}_q^L$ and denoted by rot$(t; c_1, c_2, \ldots, c_L)$ if its entries are defined as following*

$$A[i,j] = \begin{cases} c_i & \text{if } j = (i+t) \mod L \\ 0 & \text{otherwise.} \end{cases}$$

Observe that the product of two rotation matrices is again a rotation matrix. More precisely we have the following lemma (the proof is straightforward and we omit it).

**Lemma V.1.** *rot$(t_1; c_1, c_2, \ldots, c_L) \times$ rot$(t_2; b_1, b_2, \ldots, b_L) =$ rot$(t_1 + t_2; c_1 b_{1+t_1}, c_2 b_{2+t_1}, \ldots, c_L b_{L+t_1})$.*

As a result, taking the powers of a rotation matrix leads to a rotation matrix as well.

Recall that in our algorithms, we substitute the coding coefficients $\{X_i\}$ to be polynomials of an indeterminate $X$. From the previous lemma, if we would like the matrices $\{X_i\}$ to be rotation matrices, we can simply select the variable $X$ to be a rotation matrix, and then have the variables $\{X_i\}$ be powers of $X$; namely,

$$X_i = X^{m_i} \tag{19}$$

for some exponent $m_i$. We then need to select the exponents $\{m_i\}$ so that the polynomial $f(X_1, \ldots, X_\nu)$ does not become identically zero. We can easily do this by slightly modifying Step 1 in our algorithms.

*Alternative substitution for Step 1:* As before we employ the algorithm in [3] to find an assignment $X_i = \alpha_i$ with $f(\alpha_1, \alpha_1, \ldots, \alpha_\nu)$ invertible. Let $\alpha$ be a primitive element of the finite field $\mathbb{F}_q$, we can then express each $\alpha_i$ in $\mathbb{F}_q$ as $\alpha_i = \alpha^{m_i}$ for some $m_i$. Therefore we have:

$$0 \neq f(\alpha_1, \alpha_2, \ldots, \alpha_\nu) = f(\alpha^{m_1}, \alpha^{m_2}, \ldots, \alpha^{m_\nu}) = f(\alpha).$$

Selecting these values for the exponents in (19) concludes the first step.

The following, easy to prove, lemma, gives the characteristic polynomial of a rotation matrix.

**Lemma V.2.** *The characteristic polynomial of the matrix rot*$(1; c_1, c_2, \dots, c_L)$ *is* $X^L - \prod_{i=1}^{L} c_i$.

Restricting the coding matrices to be rotation matrices may not always lead to a solution; the following theorem helps explore when it is possible to have such a solution.

**Theorem V.3.** *Consider creating a polynomial* $f(X)$ *by substituting* $X_i = X^{m_i}$, *for some* $m_i$, *in the polynomial* $f(X_1, \dots, X_\nu)$ *derived from the transfer matrix of a network. If we can create such an* $f(X)$ *that is co-prime with any of the polynomials* $g(X) = X^L - c$ *for some* $c \in \mathbb{F}_q$, *then we can solve the network coding problem using rotation coding matrices.*

*Proof:* If such a co-prime polynomial exists, use $X = \text{rot}(t;\ 1 \dots 1\ c)$. ■

That is, if there exists an assignment for the variables of the polynomial $(X_1, \dots, X_\nu)$ with powers of a single variable $X$ so that $f(X)$ is co-prime with a polynomial $g(X) = X^k - c$ for some $c \in \mathbb{F}_q$, then the multicast network code design is solvable with rotational coding.

We underline that our developed algorithms do not allow us in polynomial time to find such an assignment, if it exists, but only to check, if the assignment we have selected, is solvable with rotational coding. We also do not provide any guarantees under which such an assignment exists.

### *B. Permutation Matrices over* $\mathbb{F}_q$

Another class of matrices that can be employed is the class of all matrices that are associated with the powers of a fixed permutation matrix $\mathbf{A}$. A binary permutation matrix simply permutes the elements of the vector it multiplies. Randomized designs using binary permutation matrices were presented in [16]. Our approach aims to develop finite length deterministic designs.

Permutation matrices can be considered to be block diagonal matrices[4], where each block in the diagonal is a rotation matrix; this follows from the fact that each permutation can in general be written as the product of cycles. (See [34], chapter 1, page 48). This implies that the same analysis and approach as in the case of rotation matrices can be applied, with the only difference that now the characteristic polynomials of permutation matrices are products of characteristic polynomials of rotation matrices. That is, we can substitute the coding variables as powers of a single variable, and check whether, for the resulting polynomial, there exists a co-prime polynomial that is the characteristic polynomial of a permutation matrix $\mathbf{A}$.

## VI. Scalar vs. vector operation: a comparison

As we already discussed, and as Theorem VII.4 in Appendix A proves, if we can solve the scalar network coding problem for a network over a field $\mathbb{F}_{q^m}$, with any algorithm that we desire, then we are able to solve the vector network coding problem over the same network using matrices of dimension $m \times m$, i.e., using $L = m$. Therefore, binary matrices are at least as useful as finite fields, since, every solution over a finite field $\mathbb{F}_{q^L}$ can be directly translated to code designs for vector network coding using matrices in $M_L(\mathbb{F}_q)$.

Indeed, if we translate scalar to vector coding, we only employ $q^L$ of the matrices in $M_L(\mathbb{F}_q)$, while there exist $q^{L^2}$ $L \times L$ such matrices. Even if we only count the number of invertible matrices in $M_L(\mathbb{F}_q)$, it is easy to see that this number equals $\prod_{j=0}^{L-1} (q^L - q^j)$, which in turn can be lower bounded by $q^{\frac{L(L-1)}{2}} (q-1)^{\frac{L(L+1)}{2}}$. Clearly, the number of invertible matrices of size $L$ is still much larger than $q^L$.

However, in the vector coding algorithm we have developed, we have imposed restrictions on the set of matrices we are using. In particular, we have assumed that:

1) The variables $\{X_i\}$ are pairwise commuting matrices, and
2) each such variable can be expressed as a single variable polynomial.

Theorem VI.1 proves that, under the above two assumptions, if for vector coding we find a matrix $\mathbf{A}$ of size $m \times m$ such that $f(\mathbf{A})$ is invertible, and thus we can solve the vector coding problem using size $m$, we can translate this to scalar solution over a field of size $2^k$, with $k \le m$.

**Theorem VI.1.** *Consider a polynomial* $f(x)$ *with binary coefficients, and assume that there exists an* $m \times m$ *matrix* $\mathbf{A}$ *so that* $f(\mathbf{A})$ *is invertible. Then, there exists a scalar value* $\alpha$ *in a finite field of size at most* $2^m$ *so that* $f(\alpha) \neq 0$.

*Proof:* If $f(\mathbf{A})$ is invertible for some $\mathbf{A} \in M_m(\mathbb{F}_2)$, then any eigenvalue $a$ of $\mathbf{A}$ also satisfies $f(a) \neq 0$. On the other hand, since the characteristic polynomial of $\mathbf{A}$ has degree $m$, the degree of the characteristic polynomial $h(x)$ of $a$ over $\mathbb{F}_2$ is at most $m$ (see [21], proposition 1.15). If we take the field generated by $\alpha$, it is of size at most $2^m$ and it contains $\alpha$. ■

Given this theorem, one may ask, why not simply use our algorithm for scalar network code design, and translate the resulting solution to a vector solution. We believe that the vector network coding algorithm might still be interesting, at least for the following reasons:

1) *We can work with structured matrices*. Our algorithm for vector network code design may lead to coding matrices that have a desired structure, for example be rotation matrices, as we discussed in Section V.
2) *We can work with polynomials that are not irreducible*. Scalar designs always seek for a co-prime factor that is an irreducible polynomial, since, only irreducible polynomials generate finite fields. In contrast, for our vector designs, we can use polynomials that are not necessarily irreducible. We thus have a larger set of polynomials to use, that might be useful in various application. The following example VI.2 discusses such an application.

**Example VI.2.** *The goal of this example is to illustrate why working with non-irreducible polynomials might be useful.*

[4]More precisely, every permutation matrix is similar to a block diagonal matrix where each block in the diagonal is a rotation matrix. Two square matrices are called similar if they correspond to the same linear transformation but with respect to possibly different basis (see [23], chapter 2, page 44).

*Consider a sensor network application where the payload packet length is very small and fixed to be* $5$ *bits. Assume that our algorithm steps result*[5] *to the polynomial* $f(x) = x^{32} - x$. *This polynomial has as roots all elements of the field* $\mathbb{F}_{32}$, *and as factors 8 polynomials:* $x$, $x-1$, *and the* $6$ *irreducible polynomials of degree five. Our algorithm for scalar code design can provide a solution over the fields of size* $2^2$ *and* $2^3$ *but not the field of size* $2^5$. *Thus, no scalar solution can be translated to using vectors of length* $L = 5$, *resulting in suboptimal use of the available payload space. On the contrary, our algorithm for vector code design can employ the polynomial* $g(x) = (x^2 + x + 1)(x^3 + x + 1)$, *which is a polynomial of degree* $5$ *and is co-prime with* $f(x)$. *Thus, it can lead to a vector network code solution of* $L = 5$ *which optimally utilizes the available space.*

*This example leverages the fact that if there exist vector solutions of length* $k_1$ *and* $k_2$, *there also exists a vector solution of length* $k_1 + k_2$. *In contrast, a scalar solution over fields of size* $2^{k_1}$ *and* $2^{k_2}$ *does not necessarily imply a solution over a field of size* $2^{k_1+k_2}$. *This is because a field of size* $\mathbb{F}_{2^k}$ *is a subfield of* $\mathbb{F}_{2^l}$ *only if* $k$ *divides* $l$.

3) *It provides a basis for building further algorithms.* Our algorithm for vector network coding builds on several intermediate steps, that in themselves we believe could be useful for developing more general algorithms. For example, in our algorithms we restrict our attention to polynomials of a single variable. Future algorithms may consider polynomials of two variables, as we do in the following example.

**Example VI.3.** *The goal of this example is to show that use of vector network coding can potentially lead to new algorithms that achieve solutions not possible with scalar coding.*

*Consider a network with* $h = 2$ *and* $N$ *receivers, where the transfer function to one specific receiver has the form*

$$\begin{bmatrix} h_1(X_1,\ldots,X_\nu) & 0 \\ 0 & h_2(X_1,\ldots,X_\nu) \end{bmatrix}, \tag{20}$$

*for some polynomials* $h_1$ *and* $h_2$. *Assume that we build an algorithm which substitutes each* $X_i$ *as a function of two variables,* $x$ *and* $y$, *and that we end up with the transfer matrix towards this particular receiver*

$$\begin{bmatrix} F(x,y) & 0 \\ 0 & G(x,y) \end{bmatrix}. \tag{21}$$

*Assume that*

$$F(x,y) = f_4(x)\cdot f_6(x)\cdot f_7(x)\cdot f_8(x)\cdot f_9(x)\cdot f_{10}(x),$$

*where we define* $f_i(x)$ *to be the product of all binary irreducible polynomials of degree* $i$, *and*

$$G(x,y) = ((x^4+x)(x^8+x)+(y^4+y)(y^8+y))(x^{32}+x+y^{32}+y).$$

*We want to select values for* $x$ *and* $y$ *so that this receiver has a full rank set of equations to solve*[6]. *We will show that, it is not possible to do that using scalar network coding over* any *field of size smaller or equal to* $2^{10}$; *however, there exists a vector coding solution of size* $L = 10$.

*Let* $h(x,y) = F(x,y)G(x,y)$. *We first show that the evaluation of* $h$ *at any element of the fields* $\mathbb{F}_2, \mathbb{F}_4, \ldots, \mathbb{F}_{1024}$ *is zero. If* $x, y \in \mathbb{F}_4$ *then both* $x^4 + x$ *and* $y^4 + y$ *are zero and therefore* $h(x,y) = 0$. *A similar argument shows that if* $x, y \in \mathbb{F}_8$ *or* $x, y \in \mathbb{F}_{32}$ *then* $h(x,y) = 0$. *If* $x, y \in \mathbb{F}_{16}$ *then either both of* $x$ *and* $y$ *will be in* $\mathbb{F}_4$ *or at least one of them is not in that field. If both of* $x$ *and* $y$ *are in* $\mathbb{F}_4$ *then both* $x^4 + x$ *and* $y^4 + y$ *are zero and therefore* $h(x,y) = 0$. *If one of them, say* $x$, *is in* $\mathbb{F}_{16}$ *but not in* $\mathbb{F}_4$ *then the characteristic polynomial of* $x$ *will be an irreducible polynomial of degree 4 and therefore* $f_4(x) = 0$. *Thus, in both cases,* $h(x,y) = 0$. *A similar argument shows that if* $x, y \in \mathbb{F}_{2^i}$ *for* $i = 1, 2, \ldots, 10$ *then* $h(x,y) = 0$.

*Next, we will show that* $h(x = X_1, y = X_2)$ *is an invertible matrix, where*

$$X_1 = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix} \text{ and } X_2 = \begin{bmatrix} A_3 & 0 & 0 \\ 0 & A_1 & 0 \\ 0 & 0 & A_2 \end{bmatrix},$$

*where*

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

*and*

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

*First notice that the characteristic polynomials of both matrices* $X_1, X_2$ *is the product of the characteristic polynomials of the matrices* $A_1$, $A_2$, *and* $A_3$ *which is* $c(x) = (x^2+x+1)(x^3+x+1)(x^5+x^2+1)$. *All the three factors of* $c(x)$ *are irreducible polynomials. Since* $c(X_1) = c(X_2) = 0$ *and* $c(x)$ *is co-prime with the polynomials* $f_4, f_6, f_7, f_8, f_9, f_{10}$, *then* $F(x,y)$ *will be invertible if evaluated at* $x = X_1, y = X_2$. *Thus we only need to check the invertibility of the factors* $((x^4+x)(x^8+x)+(y^4+y)(y^8+y))$ *and* $(x^{32}+x+y^{32}+y)$ *of* $G(x,y)$. *We can evaluate each of these factors to verify that all of them become invertible matrices. Therefore* $h(X_1, X_2)$ *is a* $10 \times 10$ *invertible binary matrix, and we have a vector solution that employs* $L = 10$.

Our final simple example again makes the point that, the set of vector solutions can be larger than the set of scalar solutions.

**Example VI.4.** *Consider the deterministic network in Fig. 1; deterministic networks were recently proposed to capture wireless network properties, such as broadcasting and interference [9]. The algebraic framework can be directly applied to deterministic networks as well, as observed in [13], [14], [18].*

[5]We do not claim that there exists a network solvable only if this polynomial is nonzero. We only assume that our algorithm results in this polynomial, starting from some multivariate network transfer matrix.

[6]We do not claim the existence of a network where satisfying these conditions is necessary and sufficient.

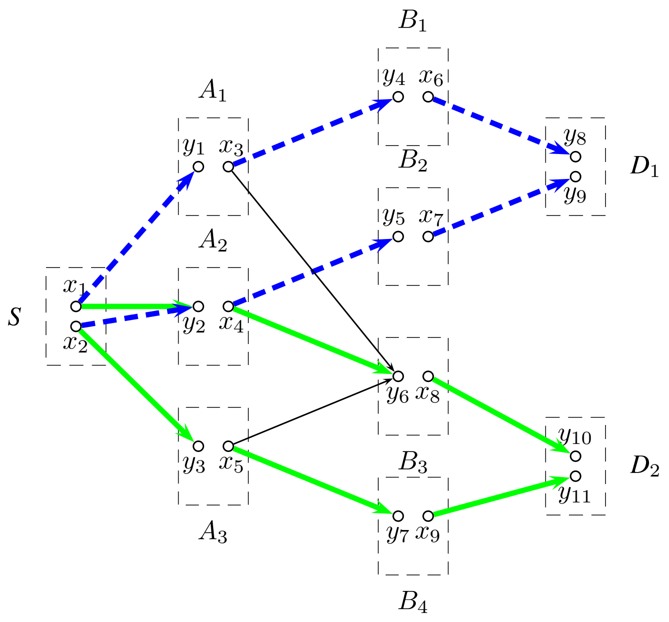

Fig. 1. A deterministic network with a source $S$ multicasting information at rate $h = 2$ to $N = 2$ receivers. The information flow to the first receiver is depicted in dashed lines, and the information flow to the second receiver in bold.

*Assume for simplicity that nodes $A_1$, $A_2$ and $A_3$ employ the coding matrices $X_1$, $X_2$ and $X_3$, while no other node employs coding. Then*

$$\mathbf{M}_1 = \begin{bmatrix} X_1 & \mathbf{0} \\ X_2 & X_2 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} X_1 + X_2 & X_2 + X_3 \\ \mathbf{0} & X_3 \end{bmatrix}.$$

*We thus require that the matrices $X_1$, $X_2$, $X_3$ and $X_1 + X_2$ are full rank. This is not possible for $L = 1$. For $L = 2$, we can employ*

$$X_1 = \mathbf{I}, \quad X_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } X_3 = \mathbf{I}. \tag{22}$$

*Consider now the vector algebraic formulations (2) and (3). We are seeking commutative matrices $X_1$, $X_2$ and $X_3$, so that the polynomial*

$$f(X_1, X_2, X_3) = X_1X_2X_3(X_1 + X_2)$$

*evaluates to an invertible $L \times L$ matrix. Note that the solution in (22) employs such matrices.*

*Next we makes the point that the set of vector solutions can be larger than the set of scalar solutions. Consider again the network in Fig. 1. The solution provided in (22) employs commutative matrices. However, we can also employ non-commutative matrices such that $X_1$, $X_2$, $X_3$ and $X_1 + X_2$ are full rank, for example*

$$X_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } X_3 = \mathbf{I}. \tag{23}$$

*We will count the number of assignments for the variables $X_1, X_2, X_3$ using $2 \times 2$ binary matrices so that the matrix $X_1X_2X_3(X_1 + X_2)$ is invertible. It suffices to select $X_3$ to be an arbitrary invertible matrix and matrices $X_1$, $X_2$ so that $X_1$, $X_2$, $X_1 + X_2$ are all invertible. Clearly there are 6 choices for $X_3$. The only way we can find two invertible binary matrix whose sum is also invertible is to choose both matrices from any of the following two sets*

$$\{\mathbf{I}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\}, \{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\}. \tag{24}$$

*This shows that there are 12 possibilities to chose the ordered pair $X_1, X_2$ satisfying the properties mentioned above. So, in total there are 72 different assignments for the $X_i$'s which make $X_1X_2X_3(X_1 + X_2)$ an invertible matrix. Now if we want to ensure that all the matrices $X_1, X_2, X_3$ are commuting with each other, we can still choose $X_1, X_2$ as before. However, given $X_1, X_2$ there are only 3 possibilities for $X_3$. Therefore there will be 36 different assignments. We can thus double the solution space if we do not restrict our attention to commutative matrices.*

## *Benefits of vector coding*

To summarize, we have argued that designing vector coding solutions, as compared to scalar solutions, has the following benefits:

- All scalar solutions over a field of size $q = 2^k$ can be translated to a vector solution of length $k$ (see Theorem VII.4).
- There may exist vector solutions of length $k$, while there exists no scalar solution in a field of size $q \leq 2^k$ (see example VI.3). In general, if we are not restricted to use commutative matrices that are polynomials of the same matrix, there exists a larger set of solutions with vector as compared to scalar coding (see example VI.4).
- If there exist vector solutions of length $k_1$ and $k_2$, there exists a vector solution of length $k_1 + k_2$. In contrast, a solution over fields of size $2^{k_1}$ and $2^{k_2}$ does not necessarily imply a solution over a field of size $2^{k_1+k_2}$ (see example VI.2).
- Vector coding allows to leverage well-studied structure and properties of matrices (see section V).

## VII. Conclusions

In this paper we develop new algebraic algorithms for the problem of vector and scalar network code design. We start our work by extending the algebraic framework developed for multicasting over graphs in [1] to include operations over matrices. Within this framework, the main idea in our approach is to reduce the problem of code design to an algebraic problem of finding co-prime factors of a given polynomial. Based on this, we provide algorithms for scalar coding that attempt to minimize the alphabet size and show a doubly exponential convergence to a solution. We also provide algorithms for vector coding that allow to use finite lengths and systematically design vector coding solutions. We illustrate how our approach can utilize structured matrices to further reduce the encoding complexity. We also offer a comparison between vector and scalar network coding and identify potential benefits of vector network coding. We believe that the approach we have developed may be useful for network optimization problems that allow coding, and for deriving additional algorithms for network code design.

## References

[1] R. Köetter and M. Médard, "Beyond routing: an algebraic approach to network coding", *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782-796, 2003.

[2] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction", *IEEE Transactions on Information Theory,* vol. 51, no. 6, pp. 1973–1982, October 2005.

[3] N. Harvey, "Deterministic network coding by matrix completion", MS Thesis 2005.

[4] T. Ho, R. Köetter, M. Médard, M. Effros, J. Shi and D. Karger, "A random linear network coding approach to multicast", *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413-4430, October 2006.

[5] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes", *IEEE International Conference on Communications (ICC)*, pp. 661–666, vol. 2, June 2004.

[6] C. Fragouli and EŠoljanin, "Decentralized network coding", *Information Theory Workshop*, 2004.

[7] R. Ahlswede, N. Cai, S-Y. R. Li and R.W. Yeung, "Network information flow", *IEEE Transactions on Informormation Theory,* vol. 46, pp. 1204–1216, July 2000.

[8] S-Y.R. Li, R.W. Yeung and N. Cai, "Linear network coding", *IEEE Transactions on Informormation Theory,* vol. 49, pp. 371–381, February 2003.

[9] S. Avestimehr, S. N. Diggavi and D. N. C. Tse, "Wireless network information flow: a deterministic approach", $arXiv$ : 0906.5394 June 2009, to appear in *IEEE Transactions on Information Theory* in 2010.

[10] A. Amaudruz and C. Fragouli, "Combinatorial algorithms for wireless information flow", *SODA*, January 2009.

[11] J. Ebrahimi B. and C. Fragouli, "Combinatorial algorithms for wireless information flow", $arXiv$ : 0909.4808, under submission, Sept. 2009.

[12] C. Shi and A. Ramamoorthy, "Fast algorithms for finding unicast capacity of linear deterministic wireless relay networks", $arXiv$ : 0909.5507, under submission, 2010.

[13] J. Ebrahimi B. and C. Fragouli, "Multicasting algorithms for deterministic networks", *IEEE Information Theory Workshop (ITW)*, Cairo, January 2010.

[14] J. Ebrahimi B. and C. Fragouli, "Vector network coding algorithms", *ISIT* 2010.

[15] J. Ebrahimi B. and C. Fragouli, "Vector network coding", *EPFL Technical Report*, January 2010.

[16] S. Jaggi, Y. Cassuto and M. Effros, "Low complexity encoding for network codes", *ISIT* 2006.

[17] M. Khojastepour and A. Keshavarz-Haddad, "Rotational coding achieves multicast capacity of deterministic wireless networks", *IEEE Allerton*, September 2009.

[18] M. Kim and M. Médard "Algebraic network coding approach to deterministic wireless relay networks", 2010, $arXiv$ : 1001.4431

[19] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 829–848, March 2006.

[20] A. Lehman and E. Lehman, "Complexity classification of network information flow problems", *ACM SODA*, 2004.

[21] P. Morandi, "Field and Galois Theory", *Springer*, 1996.

[22] I. Kovacs, D. S. Silver and S. G. Williams, "Determinants of commuting-block matrices", *The American Mathematical Monthly*, vol. 106, no. 10, pp. 950-952, December 1999.

[23] R.A. Horn and C.R. Johnson, "Matrix Analysis", *Cambdrige University Press*, 1990.

[24] I. Chaparlinski, "On finding primitive roots in finite fields", *Theoretical computer science*, vol. 157, no 2, pp. 273-275, May 1996.

[25] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities", *Journal of the ACM*, vol. 27, no. 4, pp. 701-717, 1980.

[26] J. Rimas, "On computing of arbitrary positive integer powers for one type of even order symmetric circulant matrices", *Applied Mathematics and Computation*, vol. 172, pp. 86-90, 2006.

[27] C. Fragouli and E. Soljanin, "Network coding: Fundamentals," *Foundations and Trends in Networking*, vol. 2, pp. 1–133, 2007.

[28] T.M. Cover and J.A. Thomas, "Elements of Information Theory", *John Wiley and Sons Ltd*, 2006.

[29] T. Ho, R. Köetter, M. Médard, M. Effros, J. Shi and D. Karger, "A random linear network coding approach to multicast",*IEEE Transactions on Information Theory*, vol. 52, iss. 10, pp. 4413-4430, October 2006.

[30] E. Erez, Y. Xu and E. Yeh, "Coding for the deterministic network model", *ITA*, 2010.

[31] M. Médard, M. Effros, T. Ho and D. Karger, "On coding for non-multicast networks", *Allerton*, October 2003.

[32] Ph. Piret, "On the number of divisors of a polynomial over GF(2)", *Lecture Notes in Computer Science*, vol. 228, 1984.

[33] C.H. Papadimitriou, "Computational Complexity", *John Wiley and Sons Ltd*, 2003.

[34] N. Jacobson,"Basic algebra I (Second ed.)", *Dover Publications*, 2009.

[35] S. Jaggi, M. Effros,T.C. Ho and M. Médard, "On linear network coding", *Allerton*, 2004.

[36] A.I. Barbero and D. O. Ytrehus, "An efficient centralized binary multicast network coding algorithm for any cyclic network", $arXiv$ : 0705.0085, May 2007.

[37] R.M. Gray, "Toeplitz and Circulant Matrices, A Review", *Foundations and Trends in Communications and Information Theory*, 2005.

## Appendix A

### Lemmas used in the proof of Theorem IV.3

For the proof in Theorem IV.3, we need to find a good approximation for the summation of the degrees of all irreducible polynomials of degree at most $m$. To do so, we use the following well-known result on the number of irreducible binary polynomials of a certain degree.

**Lemma VII.1.** *The number of irreducible polynomials of degree $m$ is equal to $\frac{1}{m}\sum_{d|m}\mu(\frac{m}{d})2^d$ where $\mu(d)$ is the Möbius function (see [34], chapter 4, page 289, Corollary 2). Moreover, for every $\epsilon > 0$, if $m \geq 2\log_2\frac{2}{\epsilon}$, then the number of binary irreducible polynomials of degree $m$ is at least $\frac{(1-\epsilon)}{m}2^m$.*

To see why the second statement is true, we use the fact that the number of binary irreducible polynomials of degree $m$ is lower bounded by $\frac{2^m-2\times 2^{m/2}}{m}$ [32]. It is now straightforward to check that if $m \geq 2\log_2\frac{2}{\epsilon}$ then $\frac{(1-\epsilon)}{m}2^m \leq \frac{2^m-2\times 2^{m/2}}{m}$.

**Lemma VII.2.** *Let $\epsilon$ be a positive real number and let $N_\epsilon = 2\log_2\frac{2}{\epsilon}$. If $m > 2N_\epsilon$ then the summation of the degrees of all the irreducible binary polynomials of degree at most $m$ is at least $(1-2\epsilon)2^{m+1}$.*

*Proof:* For $k = N_\epsilon, N_\epsilon + 1, \ldots, m$, the number of irreducible polynomials of degree $k$ is at least $\frac{(1-\epsilon)}{k}2^k$ by the previous lemma. Each such polynomial has degree $k$, and therefore, the summation of all the irreducible polynomials of degree $k$, equals $k$ times their number, which is at least $(1-\epsilon)2^k$. Summing up this quantity for $k = N_\epsilon, N_\epsilon+1, \ldots, m$, we see that the summation of the degrees of all the irreducible polynomials of degree at most $m$ is at least $(1-2\epsilon)2^{m+1}$. ■

We observe that the numbers $N_\epsilon$ are not large. In fact the upper bound $\frac{1}{m}2^m$ for the number of irreducible polynomials of degree $m$ is very close to the actual number. We will use in this paper this approximation for the number of irreducible polynomials, as it does not affect the order arguments. We will thus dispense of $\epsilon$ and $N_\epsilon$ and say that the summation of all the irreducible polynomials of degree at most $m$ is approximately $2^{m+1}$.

## Efficient time calculation of a polynomial determinant

**Lemma VII.3.** *Consider a $k \times k$ matrix whose elements are either binary variables or polynomials in the same variable $x$ of degree at most $p$. Then we can calculate the determinant by solving a $(pk+1) \times (pk+1)$ set of linear equations.*

*Proof:* Let $g(x)$ denote the polynomial resulting from the determinant, we know that the degree $\nu$ of the polynomial $g(x)$ is at most $\nu \leq pk$. Since we know that

$$g(x) = c_\nu x^\nu + \ldots + c_1 x + c_0$$

for some unknown coefficients $\{c_i\}$, we can simply evaluate the value of the polynomial at $\nu + 1$ points $\{\beta_i\}$ and solve a system of linear equations to retrieve $\{c_i\}$. We select $\beta_i$ from a finite field $\mathbb{F}_{q'}$ of size $q'$ much larger than $\nu$ we take the $\nu + 1$ distinct nonzero elements $\beta_j$ of $\mathbb{F}_{q'}$, and for each one, compute det$(\mathbf{M}(\beta_j))$. Notice that in the system of linear equations, the corresponding matrix is a Vandermonde matrix and in particular, it is invertible. So, there exists a unique solution for this system of linear equation.

Note that if we expand the determinant as in the definition of the determinant of a matrix, since all the entries are binary polynomials, the determinant will be a binary polynomial as well. Thus we know that $\{c_i\}$ are binary coefficients, and, although we solve linear equations over the finite field where the $\beta_i$'s belong, the unique solution of this system of equations will be binary. ■

## Mapping of scalar operations to vector operations for our algorithm

This is a well known result in algebraic coding, but we repeat it here for completeness.

**Theorem VII.4.** *For a non-zero polynomial $f(x_1, x_2, \ldots, x_n)$, assume that there are values $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_{2^m}$ with $f(\alpha_1, \alpha_2, \ldots, \alpha_n) \neq 0$. Then there are pairwise commuting matrices $A_1, A_2, \ldots, A_n \in M_m(\mathbb{F}_2)$ such that $f(A_1, A_2, \ldots, A_n)$ is an invertible matrix.*

*Proof:* Consider the natural one-to-one ring homomorphism $\phi : \mathbb{F}_{2^{\alpha(f)}} \rightarrow M_{\alpha(f)}(\mathbb{F}_2)$ as defined in [34], section 7, page 424. Let $\alpha_1, \alpha_1, \ldots, \alpha_n \in \mathbb{F}_{2^m}$ with $g(\alpha_1, \alpha_1, \ldots, \alpha_n) \neq 0$. Let $A_i := \phi(\alpha_i)$ for $1 \leq i \leq n$. Since $\phi$ is a ring homomorphism we have: $f(A_1, A_2, \ldots, A_n) = f(\phi(\alpha_1), \phi(\alpha_1), \ldots, \phi(\alpha_n)) = \phi(f(\alpha_1, \alpha_1, \ldots, \alpha_n)) = \phi(\alpha)$. Since $\alpha \neq 0$, $\phi(\alpha)$ is invertible and hence, $f(A_1, A_2, \ldots, A_n)$ is invertible. ■

## Biographies

*Javad Ebrahimi B.*

Javad Ebrahimi B. was born in 1981. He received his B.Sc degree in Pure Mathematics from Sharif University of Technology, Tehran, Iran in 2004. He started his graduate studies in 2006 and received M.Sc degree in Pure Mathematics from Simon Fraser University, Vancouver, Canada in 2008. He is currently a PhD student at the School of Computer and Communication Sciences, EPFL, Switzerland.

*Christina Fragouli*

Christina Fragouli is an Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, in 1996, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1998 and 2000, respectively. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, and DIMACS, Rutgers University. From 2006 to 2007, she was an FNS Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland.

Her research interests are in network information flow theory and algorithms, network coding, wireless sensor networks, connection s between communications, networking and computer science. She received the Fulbright Fellowship for her graduate studies, the Outstanding Ph.D. Student Award 2000-2001, UCLA, Electrical Engineering Department, the Zonta award 2008 in Switzerland, and the Young Investigator ERC grant award in 2009. She served as an editor for IEEE Communications Letters, and is currently serving as an editor for IEEE Transactions on Information Theory, IEEE Transactions on Communications and for Elsevier Computer Communications.