Finally, our approach was heavily based on the use of Shannon–Fano codes, and not on the optimal Huffman codes. The problem is much more involved and challenging when Huffman codes are considered. Pioneering work on this difficult problem has been performed by Longo [18], [19], Nemetz and Simon [20], [21]. In the papers of Longo [18], [19] the informational divergence $H(P\|Q)$ plays an important role in the problem.

## ACKNOWLEDGMENT

I wish to thank the reviewers for pointing out relationships to previous work and for providing constructive comments.

## REFERENCES

[1] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. The Univ. of Illinois, Sept. 1949.
[2] R. A. Ash, *Information Theory*. New York: Wiley-Interscience, 1965.
[3] L. D. Davisson and A. Leon-Garcia, "A source matching approach to finding minimax codes," *IEEE Trans. Inform. Theory*, vol. IT-26, no. 2, pp. 166–174, Mar. 1980.
[4] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inform. Theory*, vol. IT-19, no. 6, pp. 783–795, Nov. 1973.
[5] R. E. Krichevsky and V. K. Torfimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, no. 2, pp. 199–206, Mar. 1981.
[6] E. N. Gilbert, "Codes based on inaccurate source probabilities," *IEEE Trans. Inform. Theory*, vol. IT-17, no. 3, pp. 304–314, May 1971.
[7] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. IT-21, no. 2, pp. 194–203, Mar. 1975.
[8] J. Rissanen, "Minimax codes for finite alphabets," *IEEE Trans. Inform. Theory*, vol. IT-24, no. 3, pp. 389–392, May 1978.
[9] R. A. Ash, *Information Theory*. New York: Wiley-Interscience, 1965.
[10] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Res. Dev.*, vol. 20, pp. 197–300, May 1976.
[11] G. G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Commun.*, vol. COM-29, no. 6, pp. 858–867, June 1981.
[12] J. F. Hayes and R. R. Boorstyn, "Delay and overhead in the encoding of data sources," *IEEE Trans. Commun.*, vol. COM-29, pp. 1678–1683, Nov. 1981.
[13] B. Ya. Ryabko, "Comments on 'A source matching approach to finding minimax codes'," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 780–781, Nov. 1981.
[14] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.
[15] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton University, 1970.
[16] D. G. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.
[17] D. Levine and M. Tribus, Eds., *The Maximum Entropy Formalism*. Cambridge, MA: MIT, 1979. (Ch. by E. T. Jaynes.)
[18] G. Longo, "Informational divergence and Huffman coding," presented at the 1979 IEEE International Symposium on Information Theory, June 1979, Grignano, Italy, and 8th Prague Conference on Information Theory, 1978.
[19] G. Longo and G. Galasso, "An application of informational divergence to Huffman codes," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 1, pp. 36–42, Jan. 1982.
[20] T. Nemetz, "On the word length of Huffman codes," in *Probl. Contr. Inform. Theory*, vol. 9, no. 4, pp. 231–242, 1980.
[21] T. Nemetz and J. Simon, "Self information and optimal codes," in *Col. Math. Society Janos Bolyai*, vol. 16, *Topics in Information Theory*, I. Csiszaz and P. Elias, Eds. Keszthely, 1975, pp. 457–468.

# A New Structured Design Method for Convolutions over Finite Fields, Part I

MEGHANAD D. WAGH, MEMBER, IEEE, AND SALVATORE D. MORGERA, MEMBER, IEEE

***Abstract*—The structure of bilinear cyclic convolution algorithms is explored over finite fields. The algorithms derived are valid for any length not divisible by the field characteristic and are based upon the small length polynomial multiplication algorithms. The multiplicative complexity of these algorithms is small and depends on the field of constants. The linear transformation matrices $A$, $B$ (premultiplication), and $C$ (postmultiplication) defining the algorithm have block structures which are related to one another. The rows of $A$ and $B$ and the columns of $C$ are maximal length recurrent sequences. Because of the highly regular structure of $A$, $B$, and $C$, the algorithms can be very easily designed even for large lengths. The application of these algorithms to the decoding of Reed–Solomon codes is also examined.**

Manuscript received June 30, 1980; revised August 20, 1982. This work was supported by NSERC Grant A0912. This work was partially presented at the 1981 IEEE International Symposium on Information Theory, Santa Monica, CA, February 9–12.

M. D. Wagh is with Old Dominion University, Department of Electrical Engineering, Norfolk, VA 23508.

S. D. Morgera is with Concordia University, Department of Electrical Engineering, 1455 de Maisonneuve Blvd. W., Montréal, H3G 1M8, PQ, Canada.

## I. INTRODUCTION

CYCLIC convolution of discrete sequences plays an important role in signal processing. It is useful for describing the output of a linear-time-invariant finite impulse response system to a periodic input, and it is the basis for computing other necessary signal processing results. Furthermore, linear convolution, important for pre-

0018-9448/83/0700-0583$01.00 

dicting system responses to nonperiodic inputs, polynomial products, and large integer products, may be computed through cyclic convolutions. Signal processing transforms and, in particular, the discrete Fourier transform can also be calculated via cyclic convolution.

Consequently a large number of papers dealing with cyclic convolution of real and complex sequences have recently appeared [1]–[14]. By comparison, convolution of finite field sequences, though equally important, has received quite a bit less attention. Early approaches to computational algorithms for cyclic convolution were based upon two facts: 1) If the length $N$ of the sequences can be factored in mutually prime factors $n_1, n_2, \cdots, n_k$, then the cyclic convolution can be computed as a $k$-dimensional cyclic convolution with length in the $i$th dimension being $n_i$. 2) The algorithm of a factor length $n_i$ may be obtained by viewing the convolution as a product of two polynomials in $z$ modulo $z^{n_i} - 1$ and evaluating it by first finding the partial products modulo each factor of $z^{n_i} - 1$ and then combining these using the Chinese remainder theorem to get the required convolution.

The design of an efficient algorithm (i.e., one with a small computational complexity) of length $n_i$ is quite complicated and time consuming. The strategy normally adopted is to design good algorithms as per fact 2) only for a few small $n_i$ values and combine these small length algorithms using multidimensional techniques to obtain algorithms for longer lengths. This approach, however, has some disadvantages. The only efficient algorithms known in the literature [1] are for lengths between 2 and 9.[1] Based on these, a total of only 47 algorithms may be constructed using multidimensional techniques. This means that the application of cyclic convolution has to be tailored to fit one of these available lengths. In addition, the small length algorithms are efficient only over the complex number field and though sometimes it is possible to modify them to obtain convolution over the required finite field, the modified algorithm may not be efficient. Finally, the algorithms obtained in this fashion have very little regular structure which may be exploited in the designing of efficient hardware or software. More recent efforts to introduce some regularity into the algorithm has led to the discovery of polynomial transforms [12]–[14]. Though algorithms obtained in this manner are a little more regular than the earlier Agarwal–Cooley algorithms [1], they still lack a completely regular structure.

In this paper we present a new approach to obtain a length $N$ cyclic convolution bilinear algorithm over the field $\mathrm{GF}(p^m)$, where $p$ is a prime and $m$ is an arbitrary positive integer, for any $N$ not divisible by $p$. We show that when constrained with only the multiplicative complexity, as is generally the case with finite field operations, it is possible to design the algorithm very quickly by using the principles outlined in this paper. The approach, we feel, is the only *systematic structured design method* available for this type of algorithm development.

[1]This statement is not strictly true if one also is using convolution algorithms based on number theoretic transforms.

The information about the field of constants is incorporated in the design of the algorithm. Consequently, the multiplicative complexity of the algorithm depends not only on its length, but also on the field of constants. This direct approach yields much more efficient algorithms than those obtained by modifying algorithms designed for the complex number field. The algorithm obtained here is bilinear in nature and the matrices defining it are *very highly structured.*

This paper is organized in the following manner. Section II develops the necessary mathematical background. Actual design of the algorithm is discussed in Sections III and IV. These designs are illustrated through examples in Section V and their computational complexity is examined in Section VI. Finally, Section VII demonstrates the application of the algorithms developed for the decoding of Reed–Solomon codes.

## II. Mathematical Preliminaries

The mathematical preliminaries fundamental to the content of this work are presented in this section. In particular, we show that a bilinear algorithm with field of constants $\mathrm{GF}(p)$ designed for input data over $\mathrm{GF}(p)$ also works when the input data is over $\mathrm{GF}(p^m)$. We also introduce a certain partitioning of integer sets which gives rise to the algorithm in the form presented here.

*Lemma 1:* A bilinear algorithm (with field of constants $\mathrm{GF}(p)$) which is valid for input data over $\mathrm{GF}(p)$ is also valid for input data over $\mathrm{GF}(p^m)$.

*Proof:* Consider the following algorithm to evaluate the bilinear form $w = u * v$ when $u$ and $v$ are vectors over $\mathrm{GF}(p)$: $w = C(Au \times Bv)$, where $A$, $B$, and $C$ are matrices of appropriate dimensions over $\mathrm{GF}(p)$ and $\times$ denotes a component-by-component multiplication of vectors. We now prove that $w$ can be computed in the same manner even if $u$ and $v$ are vectors over $\mathrm{GF}(p^m)$. To show this, note that in this case, we can express $u$ and $v$ as

$$u = \sum_{i=0}^{m-1} u_i \alpha^i, \qquad v = \sum_{j=0}^{m-1} v_j \alpha^j,$$

where $u_i$ and $v_j$ are vectors over $\mathrm{GF}(p)$ and $\alpha$ is the primitive element of $\mathrm{GF}(p^m)$. Then using bilinearity, we obtain

$$w = \left( \sum_{i=0}^{m-1} u_i \alpha^i \right) * \left( \sum_{j=0}^{m-1} v_j \alpha^j \right) = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \left( \alpha^{i+j} u_i * v_j \right).$$

Further, since $u_i$ and $v_j$ are over $\mathrm{GF}(p)$, we may use the algorithm to compute $u_i * v_j$. Thus

$$\begin{aligned} w &= \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \alpha^{i+j} C\left( Au_i \times Bv_j \right) \\ &= C\left[ \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \left( Au_i \alpha^i \times Bv_j \alpha^j \right) \right] \\ &= C\left[ \left( A \sum_{i=0}^{m-1} u_i \alpha^i \right) \times \left( B \sum_{j=0}^{m-1} v_j \alpha^j \right) \right] = C(Au \times Bv). \end{aligned}$$

Thus, the same algorithm can be used to compute $u * v$ even when $u$ and $v$ are vectors over GF$(p^m)$.

The important notion of *cyclotomic sets* is now introduced. Let $N$ be of the form $p^n - 1$. Partition the integers 0 through $N-1$ into sets $S_{i_1}, S_{i_2}, S_{i_3}, \cdots$. A set $S_i$ is generated by starting from the smallest integer $i$ not covered in an earlier set and defining its other members as

$$S_i = \{i, ip \bmod N, ip^2 \bmod N, ip^3 \bmod N, \cdots\}.$$

Since $ip^n \bmod N = i$, each set $S_i$ is finite. We denote the number of elements in $S_i$ by $\sigma_i$ and the set of indices $\{i_1, i_2, i_3, \cdots\}$ by $\mathbb{S}$. We list below the properties of the $\sigma_i$ which will be used later. Proofs of these properties are given in Appendix I.

P1) $0 \in \mathbb{S}$ and $\sigma_0 = 1$.
P2) Given any divisor $t$ of $n$ (including $t = 1$ when $p \neq 2$), $\theta = (p^n - 1)/(p^t - 1) \in \mathbb{S}$ and $\sigma_\theta = t$.
P3) For any nonzero $i \in \mathbb{S}$, it is possible to find a $\theta \in \mathbb{S}$ having the form given in P2) for some $t$, such that $i$ is a multiple of $\theta$ and $\sigma_i = \sigma_\theta$.
P4) With $\theta$ as in P2) and $\alpha$, a primitive element of GF$(p^n)$, $\alpha^\theta$ generates the subfield GF$(p^{\sigma_\theta})$ of GF$(p^n)$ and, consequently, $\sigma_\theta | n$.
P5) From P3) and P4), for any $i \in \mathbb{S}$, $\sigma_i | n$ and $\alpha^i \in$ GF$(p^{\sigma_i})$.

The following example illustrates the notion of cyclotomic sets.

*Example:* With $p = 2$ and $N = 2^4 - 1 = 15$, we have

$$\begin{aligned} S_0 &= \{0\}, & \sigma_0 &= 1 \\ S_1 &= \{1,2,4,8\}, & \sigma_1 &= 4 \\ S_3 &= \{3,6,12,9\}, & \sigma_3 &= 4 \\ S_5 &= \{5,10\}, & \sigma_5 &= 2 \\ S_7 &= \{7,14,13,11\}, & \sigma_7 &= 4; \quad \mathbb{S} = \{0,1,3,5,7\}. \end{aligned}$$

Note that since $n = 4$, its only divisors are 2 and 4 and $\theta$ corresponding to these as in P2) are 5 and 1, respectively. Both of these integers belong to $\mathbb{S}$. The other nonzero members 3 and 7 of $\mathbb{S}$ are both multiples of 1 and $\sigma_3 = \sigma_7 = \sigma_1$ as indicated by P3).

## III. Algorithm Construction

The cyclic convolution of sequences $u$ and $v$ of length $N$ is a sequence $w$ defined as

$$w(k) = \sum_{i=0}^{N-1} u(i)v((k-i) \bmod N), \qquad 0 \leq k \leq N-1$$

and denoted by $w = u * v$. In this section we design a bilinear algorithm to compute this cyclic convolution when the data sequences are over GF$(p^m)$, the field of constants is GF$(p)$, and the length $N = p^n - 1$ for some integer $n$. By the reasoning given in Section II, it is sufficient to assume that the components of $u$ and $v$ are also from GF$(p)$.

It is then well known that $w$ can be obtained from $U$ and $V$, the Fourier transforms of $u$ and $v$ in GF$(p^n)$ as

$$w(k) = (1/N) \sum_{j=0}^{N-1} U(j)V(j)\alpha^{-jk}, \qquad 0 \leq k \leq N-1,$$

where $\alpha$ is a primitive element of GF$(p^n)$. We choose to carry out this summation first over the elements of a set $S_i$ to give $w_i(k)$ and then over all such sets. Thus,

$$w(k) = \sum_{i \in \mathbb{S}} w_i(k), \tag{1}$$

where

$$w_i(k) = (1/N) \sum_{j \in S_i} U(j)V(j)\alpha^{-jk}. \tag{2}$$

We now show that by using bilinear small degree polynomial multiplication algorithms (some of which are given in Appendix II), we can design bilinear algorithms over GF$(p)$ for (2).

*Theorem 1:* The vector $w_i$ can be computed from the $u$ and $v$ vectors by a bilinear algorithm over GF$(p)$.

*Proof:* When $i = 0$,

$$w_0(k) = (1/N)\left(\sum_{i=0}^{N-1} u(i)\right)\left(\sum_{j=0}^{N-1} v(i)\right) = C_0(A_0 u \times B_0 v),$$

where $A_0$, $B_0$, and $C_0$ are length $N$ vectors

$$A_0 = [1 \ 1 \ \cdots \ 1]; \quad B_0 = A_0; \quad C_0 = A_0^T/N.$$

For nonzero $i$ values, since any $j \in S_i$ can be expressed as $ip^l \bmod N$, $l = 0, 1, \cdots, \sigma_i - 1$, we have

$$U(j) = \sum_{k=0}^{N-1} u(k)\alpha^{ikp^l}, \qquad j = ip^l.$$

Note, however, that $u(k) \in$ GF$(p)$ and, consequently,

$$(u(k))^{p^l} = u(k).$$

Thus,

$$U(j) = \sum_{k=0}^{N-1} (u(k)\alpha^{ik})^{p^l} = \left(\sum_{k=0}^{N-1} u(k)\alpha^{ik}\right)^{p^l} = (U(i))^{p^l}.$$

Similarly, $V(j) = (V(i))^{p^l}$. Equation (2) now becomes

$$w_i(k) = (1/N) \sum_{l=0}^{\sigma_i - 1} (U(i)V(i)\alpha^{-ik})^{p^l}. \tag{3}$$

Note that

$$U(i) = \sum_{k=0}^{N-1} u(k)\alpha^{ik}, \tag{4}$$

and $i$ is a multiple of some $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1) \in \mathbb{S}$ from P3) of Section II. Therefore, $U(i)$ is a polynomial in $\alpha^\theta$. Moreover, from P4), the *minimal polynomial* over GF$(p)$ of $\alpha^\theta$ is of degree $\sigma_\theta$. Using the minimal polynomial to reduce the $\sigma_\theta$th and higher powers of $\alpha^\theta$, $U(i)$ can be reduced to a polynomial in $\alpha^\theta$ of degree $\sigma_\theta - 1$. Each coefficient of this polynomial is obtained by a linear combination (over GF$(p)$) of the components of $u$. Similarly,

$V(i)$ can be reduced to a polynomial in $\alpha^\theta$ of degree $\sigma_\theta - 1$ with coefficients expressed as linear combinations of the components of $v$. At this stage, one may use bilinear small degree polynomial multiplication algorithms (given in Appendix II) over GF($p$) to obtain the coefficients $r_m$ of the product polynomial $U(i)V(i)$ in $\alpha^\theta$ of degree $2\sigma_\theta - 2$. To prove the theorem, it is therefore sufficient to show that $w_i(k)$ (for any $k$) can be expressed as a linear combination of the $r_m$ coefficients over GF($p$).

Using the product polynomial in (3), one has

$$\begin{aligned} w_i(k) &= (1/N)\sum_{l=0}^{\sigma_i-1}\left(\left(\sum_{m=0}^{2\sigma_\theta-2} r_m\alpha^{\theta m}\right)\alpha^{-ik}\right)^{p^l} \\ &= (1/N)\sum_{l=0}^{\sigma_i-1}\sum_{m=0}^{2\sigma_\theta-2} r_m(\alpha^{\theta m-ik})^{p^l} \\ &= (1/N)\sum_{m=0}^{2\sigma_\theta-2} r_m \operatorname{tr}(\alpha^{\theta m-ik}), \end{aligned} \tag{5}$$

where the trace function $\operatorname{tr}(\cdot)$ [16], tr: GF($p^{\sigma_\theta}$) $\to$ GF($p$), is defined as

$$\operatorname{tr}(\beta) = \sum_{l=0}^{\sigma_i-1}\beta^{p^l}, \qquad \beta \in \mathrm{GF}(p^{\sigma_\theta}).$$

Note that since $\alpha^\theta$, $\alpha^i \in \mathrm{GF}(p^{\sigma_\theta})$, $\alpha^{\theta m-ik} \in \mathrm{GF}(p^{\sigma_\theta})$. Thus, $w_i(k)$ in (5) is always a linear combination over GF($p$) of the $r_m$'s.

Theorem 1 asserts that there exist matrices $A_i$, $B_i$, and $C_i$ over GF($p$) such that $w_i = C_i(A_iu \times B_iv)$. We then immediately obtain from (1) the bilinear algorithm over GF($p$) for $w$ as

$$w = C(Au \times Bv), \tag{6}$$

where $C = [C_{i_1}, C_{i_2}, \cdots]$, $A = [A_{i_1}^T, A_{i_2}^T, \cdots]^T$, and $B = [B_{i_1}^T, B_{i_2}^T, \cdots]^T$; $i_1, i_2, \cdots \in \mathbb{S}$.

From (6), it is clear that the key to the design of a bilinear algorithm over GF($p$) to convolve sequences of length $p^n - 1$ lies in the construction of the matrices $A_i$, $B_i$, and $C_i$ for all $i \in \mathbb{S}$. Referring to P3) of Section II, for every nonzero $i \in \mathbb{S}$ it is possible to find a $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1) \in \mathbb{S}$ such that $i$ is a multiple of $\theta$ and $\sigma_i = \sigma_\theta$. It is shown in the following theorem that for each such $i$ and $\theta$, the matrices $A_i$, $B_i$, and $C_i$ are related to matrices $A_\theta$, $B_\theta$, and $C_\theta$, respectively. Theorem 3 then discusses the structure of $A_\theta$, $B_\theta$, and $C_\theta$, allowing construction of the algorithm using (6).

*Theorem 2:* For $i$, $\theta \in \mathbb{S}$, if $\theta | i$, $\sigma_\theta = \sigma_i$ and $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1)$, then

$$\begin{aligned} A_i(\gamma, \delta) &= A_\theta(\gamma, (i/\theta)\delta \bmod (N/\theta)), \\ B_i(\gamma, \delta) &= B_\theta(\gamma, (i/\theta)\delta \bmod (N/\theta)), \\ C_i(\gamma, \delta) &= C_\theta((i/\theta)\gamma \bmod (N/\theta), \delta). \end{aligned}$$

*Proof:* The $A_i$ and $B_i$ matrices are used in the design of the algorithm (see proof of Theorem 1) to deliver the linear forms in $u(k)$'s and $v(k)$'s suitable for the multiplication of the $\sigma_\theta - 1$ degree polynomials $U(i)$ and $V(i)$ in $\alpha^\theta$. Similarly, $A_\theta u$ and $B_\theta v$ provide the linear forms required for the multiplication of the $\sigma_\theta - 1$ degree polynomials $U(\theta)$ and $V(\theta)$ in $\alpha^\theta$. Since in both the cases the polynomial degrees are the same, identical multiplication algorithms could be used. Note, however, from (4) that the coefficient of $\alpha^{ik}$ in $U(i)$ is $u(k)$, whereas, in $U(\theta)$, it is $u((i/\theta)k \bmod (N/\theta))$.[2] Thus, if $u(k)$ is replaced by $u((i/\theta)k \bmod (N/\theta))$ for all $k$ in the polynomial $U(i)$, we obtain the polynomial $U(\theta)$. Therefore, the linear forms $A_iu$ are seen to be the same as $A_\theta\bar{u}$ where $\bar{u}$ is obtained from $u$ by replacing every $u(k)$ by $u((i/\theta)k \bmod (N/\theta))$. This proves the relation between $A_i$ and $A_\theta$. Identical arguments applied to $V(i)$ and $V(\theta)$ give the relation between $B_i$ and $B_\theta$.

Finally, from (5) we have

$$\begin{aligned} &w_\theta((i/\theta)\gamma \bmod (N/\theta)) \\ &\quad= \frac{1}{N}\sum_{m=0}^{2\sigma_\theta-2} r_m \operatorname{tr}\left(\alpha^{\theta m-\theta((i/\theta)\gamma \bmod (N/\theta))}\right) \\ &\quad= \frac{1}{N}\sum_{m=0}^{2\sigma_\theta-2} r_m \operatorname{tr}(\alpha^{\theta m-i\gamma}) = w_i(\gamma), \end{aligned}$$

which immediately provides the relation between $C_i$ and $C_\theta$.

From Theorem 2, it is obvious that we need only obtain $A_\theta$, $B_\theta$, and $C_\theta$ for all $\theta \in \mathbb{S}$ of the type $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1)$. Note that for such a $\theta$, $\alpha^\theta$ is the root of a primitive polynomial of degree $\sigma_\theta$ over GF($p$). This polynomial, denoted by $P_\theta$, is given by

$$P_\theta(x) = (x - \alpha^\theta)(x - \alpha^{\theta p})(x - \alpha^{\theta p^2}) \cdots (x - \alpha^{\theta p^{\sigma_\theta - 1}})$$

when these factors are multiplied, all the coefficients in the $P_\theta(x)$ polynomial turn out to be in GF($p$). If

$$P_\theta(x) = x^{\sigma_\theta} - a_1x^{\sigma_\theta - 1} - a_2x^{\sigma_\theta - 2} - \cdots - a_{\sigma_\theta}, \tag{7}$$

then it is known that a *linear periodic recurrent sequence* $\{x_i\}$ of elements of GF($p$) with period $p^{\sigma_\theta} - 1$ can be obtained from the difference equation

$$x_i = a_1x_{i-1} + a_2x_{i-2} + \cdots + a_{\sigma_\theta}x_{i-\sigma_\theta} \tag{8}$$

with an arbitrary nonzero initial condition. This sequence is known as a *maximal length recurrent sequence* (MLRS) and exhibits pseudorandom properties. We will refer to it as a MLRS generated by $P_\theta$, since the coefficients in (8) are derived from those of $P_\theta$.

The reciprocal polynomial of $P_\theta$, denoted by $P_{-\theta}$, can be expressed as

$$\begin{aligned} P_{-\theta}(x) = (x - \alpha^{-\theta})(x - \alpha^{-\theta p}) \\ \cdot (x - \alpha^{-\theta p^2}) \cdots (x - \alpha^{-\theta p^{\sigma_\theta - 1}}). \end{aligned}$$

The reciprocal polynomial $P_{-\theta}$ also has degree $\sigma_\theta$, is primi-

[2] $U(\theta) = \sum_{k=0}^{N-1} u(k)\alpha^{\theta k}$. Suppose the coefficient of $\alpha^{ik}$ is $u(k')$. Then $\theta k' = ik \bmod N$ or $k' = (i/\theta)k \bmod (N/\theta)$.

tive, and generates a MLRS from the difference equation

$$x_i = a'_1 s_{i-1} + a'_2 x_{i-2} + \cdots + a'_{\sigma_\theta} x_{i-\sigma_\theta}, \qquad (9)$$

where the coefficients $a'_1, a'_2, \cdots, a'_{\sigma_\theta}$ are picked from the polynomial expression for $P_{-\theta}(x)$, i.e.,

$$P_{-\theta}(x) = x^{\sigma_\theta} - a'_1 x^{\sigma_\theta - 1} - a'_2 x^{\sigma_\theta - 2} - \cdots - a'_{\sigma_\theta}.$$

Theorem 3 gives the connection between the MLRS's and the $A_\theta$, $B_\theta$, and $C_\theta$ matrices.

*Theorem 3:* The rows of $A_\theta$ and $B_\theta$ are the maximal length recurrent sequences (MLRS's) generated by $P_\theta$ and the columns of $C_\theta$ are MLRS's generated by $P_{-\theta}$.

*Proof:* Consider

$$U(\theta) = \sum_{k=0}^{N-1} u(k)\alpha^{\theta k} = \sum_{j=0}^{\sigma_\theta - 1} f_j(\alpha^\theta)^j, \qquad (10)$$

where the second expression is obtained by using the fact that $\alpha^\theta$ is a root of $P_\theta$. For any $j$ $f_j$ is a linear combination (over GF($p$)) of the $u(k)$'s as the coefficients of $P_\theta$ are from GF($p$). Let

$$f_j = \sum_{k=0}^{N-1} R_j(k)u(k), \qquad R_j(k) \in \mathrm{GF}(p). \qquad (11)$$

We first show that the sequence of components of the vector $R_j = [R_j(0), R_j(1), \cdots, R_j(N-1)]$ is a MLRS generated by $P_\theta$. From (10) and (11), we have

$$\sum_{k=0}^{N-1} u(k)\alpha^{\theta k} = \sum_{k=0}^{N-1} u(k) \sum_{j=0}^{\sigma_\theta - 1} R_j(k)(\alpha^\theta)^j$$

or

$$(\alpha^\theta)^k = \sum_{j=0}^{\sigma_\theta - 1} R_j(k)\alpha^{\theta j}. \qquad (12)$$

Since $P_\theta(\alpha^\theta) = 0$, from (7) we obtain for any integer $k \geq \sigma_\theta$ that

$$(\alpha^\theta)^k = \sum_{l=1}^{\sigma_\theta} a_l(\alpha^\theta)^{k-l}. \qquad (13)$$

Combining (13) with (12) yields

$$\sum_{j=0}^{\sigma_\theta - 1} \alpha^{\theta j} R_j(k) = \sum_{l=1}^{\sigma_\theta} a_l(\alpha^\theta)^{k-l} = \sum_{j=0}^{\sigma_\theta - 1} \alpha^{\theta j} \sum_{l=1}^{\sigma_\theta} a_l R_j(k-l)$$

or

$$R_j(k) = \sum_{l=1}^{\sigma_\theta} a_l R_j(k-l).$$

Comparing with (8), it is obvious that $R_j$ is a MLRS generated by $P_\theta$.

Now $A_\theta u$ gives the linear forms in $f_j$'s used in the multiplication $U(\theta)V(\theta)$. This means that each row of $A_\theta$ is a linear combination (over GF($p$)) of the $R_j$ row vectors. Since a linear combination of maximal length recurrent sequences generated by the same polynomial is again a maximal length recurrent sequence, it follows that each row of $A_\theta$ is a MLRS generated by $P_\theta$. The proof for $B_\theta$ is similar. To prove the last part of the theorem, note from (5) that

$$w_\theta(k) = (1/N) \sum_{m=0}^{2\sigma_\theta - 2} r_m \,\mathrm{tr}\,(\alpha^{\theta m - \theta k})$$

and recall that the trace function is a linear function. Using the polynomial expression for $P_{-\theta}$ and the fact that $P_{-\theta}(\alpha^{-\theta}) = 0$, we have

$$\alpha^{-\theta\sigma_\theta} = \sum_{l=1}^{\sigma_\theta} a'_l \alpha^{-\theta(\sigma_\theta - l)} \quad \text{or} \quad \sum_{l=1}^{\sigma_\theta} a'_l \alpha^{\theta l} = 1. \qquad (14)$$

Consider the expression for any $k \geq \sigma_\theta$

$$\sum_{l=1}^{\sigma_\theta} a'_l w_\theta(k-l) = (1/N) \sum_{l=1}^{\sigma_\theta} a'_l \sum_{m=0}^{2\sigma_\theta - 2} r_m \,\mathrm{tr}\,(\alpha^{\theta m - \theta(k-l)}).$$

By using the linearity of the trace function and (14), the expression becomes

$$\sum_{l=1}^{\sigma_\theta} a'_l w_\theta(k-l) = (1/N) \sum_{m=0}^{2\sigma_\theta - 2} r_m \,\mathrm{tr}\left( \sum_{l=1}^{\sigma_\theta} a'_l \alpha^{-\theta(k-m-l)} \right)$$

$$= (1/N) \sum_{m=0}^{2\sigma_\theta - 2} r_m \,\mathrm{tr}\,(\alpha^{-\theta(k-m)}) = \omega_\theta(k),$$

which immediately shows that the columns in the $C_\theta$ matrix are MLRS's generated by the polynomial $P_{-\theta}$ through the difference (9).

To compute the elements of any row of $A_\theta$, we would generally be required to express $U(\theta)$ as a degree $\sigma_\theta - 1$ polynomial as in (10); express each $f_j$ as a linear combination of $u(k)$'s; and finally, knowing the linear forms in $f_j$'s which occur in the multiplication of polynomials of degree $\sigma_\theta - 1$ (Appendix II), find the required linear forms in $u(k)$'s by replacing each $f_j$ by an expression in $u(k)$'s.

However, an application of Theorems 2 and 3 simplifies this procedure to the following steps which constitute a new structured design method for convolutions over finite fields.

*Step 1:* Since the algorithm to be obtained is data independent, considering a particular data sequence $u(k) = 0$, $k \geq \sigma_\theta$, we can express $f_k$ as $f_k = u(k)$, $k = 0, 1, \cdots, \sigma_\theta - 1$. Thus, the linear forms in $f_k$'s involved in the multiplication of polynomials of degree $\sigma_\theta - 1$ (Appendix II) directly become the linear forms in $u(k)$'s, $k = 0, 1, \cdots, \sigma_\theta - 1$. The first $\sigma_\theta$ elements of any row of $A_\theta$ may, therefore, be immediately written down.

*Step 2:* The remaining portion of each row of $A_\theta$ is then completed by using the difference (8) based on $P_\theta$.

*Step 3:* $B_\theta$ is also designed using the same procedure.

*Step 4:* Equation (5) is evaluated for $k = 0, 1, \cdots, \sigma_\theta - 1$ to obtain the first $\sigma_\theta$ rows of $C_\theta$.

*Step 5:* Each column of $C_\theta$ is then completed by using the difference equation (9) based on $P_{-\theta}$. (The MLRS generated by $P_{-\theta}$ can also be obtained by reading backwards the MLRS generated by $P_\theta$.)

*Step 6:* Once $A_\theta$, $B_\theta$, and $C_\theta$ are obtained for all $\theta$ of the type $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1)$, the matrices $A_i$, $B_i$, and $C_i$ for other $i \in \mathbb{S}$ are obtained using Theorem 2.

*Step 7:* Finally, the matrices $A$, $B$, and $C$ used in the bilinear algorithm over GF$(p)$ for convolving $u$ and $v$ of lengths $p^n - 1$ are constructed through (6).

## IV. Convolutions of Factor Lengths

The major drawback of the algorithms developed in the earlier section seems to be the restriction of the convolution length to $p^n - 1$ when the field of constants is GF$(p)$. This is corrected to a great extent in this section where we show that essentially the same procedure, as outlined earlier, can be used to design convolution algorithms of *any length* $N$ relatively prime to $p$.

We first show that given any such $N$ (relatively prime to $p$) we can always find an integer $n > 0$ such that $N$ is a factor of $p^n - 1$. We then demonstrate the simple construction of the algorithm of length $N$ from that of length $p^n - 1$.

*Lemma 2:* If $p \nmid N$, there exists a positive integer $n$ such that $N | (p^n - 1)$.

*Proof:* Note that $p \bmod N$ belongs to the set of integers less than $N$ and relatively prime to $N$ which form a group under the operation of multiplication mod $N$. Let the order of this group be $n$. The order of any group element must then divide $n$, or $(p \bmod N)^n = 1 \bmod N$ or $p^n = 1 \bmod N$ and thus $N | (p^n - 1)$.

To construct an algorithm of length $N$, assume first that the algorithm of length $p^n - 1$ has been constructed as described in Section III. In other words, the integers $\{0, 1, 2, \cdots, p^n - 2\}$ have been partitioned into sets $S_{i_1}, S_{i_2}, \cdots$ as in Section II, $\mathbb{S}$ has been defined as $\mathbb{S} = \{i_1, i_2, \cdots\}$, and the relevant matrices $A_{i_1}, A_{i_2}, \cdots$; $B_{i_1}, B_{i_2}, \cdots$; $C_{i_1}, C_{i_2}, \cdots$; $i_1, i_2, \cdots \in \mathbb{S}$ which occur in the algorithm of length $p^n - 1$ have been constructed as in Section III.

Let $N'$ denote $(p^n - 1)/N$ and define the set $\mathbb{S}_N$ as

$$\mathbb{S}_N = \{i \in \mathbb{S} \mid N' | i\}.$$

The following theorem provides the algorithm for length $N$.

*Theorem 4:* The cyclic convolution $w$ of the sequences $u$ and $v$ of length $N$ over GF$(p^m)$ can be obtained as $w = C'(A'u \times B'v)$, where $C' = (N' \bmod p)[C'_{i_1}, C'_{i_2}, \cdots]$, $A' = [A'^T_{i_1}, A'^T_{i_2}, \cdots]^T$, and $B' = [B'^T_{i_1}, B'^T_{i_2}, \cdots]^T$; $i_1, i_2, \cdots \in \mathbb{S}_N$. The $A'_i$ and $B'_i$, $i \in \mathbb{S}_N$ are the matrices formed by the first $N$ columns of $A_i$ and $B_i$, respectively; and $C'_i$, $i \in \mathbb{S}_N$ are the matrices formed by the first $N$ rows of the $C_i$ matrix.

*Proof:* We first construct sequences $\bar{u}$ and $\bar{v}$ of length $p^n - 1$ as $\bar{u} = (u, u, \cdots, u)$ and $\bar{v} = (v, v, \cdots, v)$. Then it is simple to check that $\bar{w} = \bar{u} * \bar{v}$ is also periodic with period $N$, and that

$$\bar{w} = (N' \bmod p)w. \tag{15}$$

Further, the algorithm of length $p^n - 1$ provides

$$\bar{w} = C(A\bar{u} \times B\bar{v}), \tag{16}$$

where the terms $A\bar{u}$ and $B\bar{v}$ can be simplified as $A\bar{u} = [A_{i_1}^T, A_{i_2}^T, \cdots]^T \bar{u} = [\bar{A}_{i_1}^T, \bar{A}_{i_2}^T, \cdots]^T u$, where

$$\bar{A}_i(k, j) = \sum_{l=0}^{N'-1} A_i(k, j + lN), \qquad i \in \mathbb{S}. \tag{17}$$

Clearly, we have $\bar{A}_0 = (N' \bmod p)A'_0$. For any nonzero $i \in \mathbb{S}$, we can find a $\theta = (p^n - 1)/(p^{\sigma_\theta} - 1)$ such that $\theta | i$ from P3). From Theorem 3, the rows of $A_\theta$ are the MLRS's generated by the primitive polynomial of $\alpha^\theta$. This yields

$$A_\theta(k, j) = \operatorname{tr}(\beta_k \alpha^{\theta j}), \tag{18}$$

where $\beta_k$ is an element of GF$(p^n)$ chosen to satisfy (18) for the first $n$ values of $j$. Further, from Theorem 2, the rows of $A_i$ are obtained by sampling those of $A_\theta$ with period $(i/\theta)$ and hence

$$A_i(k, j) = \operatorname{tr}(\beta_k^{\alpha^{ij}}). \tag{19}$$

Since tr is a linear function, (17) and (19) imply

$$\bar{A}_i(k, j) = \operatorname{tr}\left(\beta_k^{\alpha^{ij}} \sum_{l=0}^{N'-1} \alpha^{ilN}\right). \tag{20}$$

If $\alpha^{iN} \neq 1$, the summation of $\alpha$ powers in (20) is

$$(\alpha^{N'iN} - 1)/(\alpha^{iN} - 1) = 0.$$

On the other hand, if $\alpha^{iN} = 1$, the same summation is $N' \bmod p$. Thus,

$$\bar{A}_i(k, j) = \begin{cases} 0, & \alpha^{iN} \neq 1; \\ (N' \bmod p)A_i(k, j), & \alpha^{iN} = 1, \end{cases}$$

or

$$\bar{A}_i = \begin{cases} 0, & \alpha^{iN} \neq 1; \\ (N' \bmod p)A'_i, & \alpha^{iN} = 1. \end{cases} \tag{21}$$

A similar observation may also be made with regard to the matrix $\bar{B}_i$. The required result then follows directly from (15)–(17) and (21) and the fact that the conditions $i \in \mathbb{S}$ and $\alpha^{iN} = 1$ together are equivalent to a single condition $i \in \mathbb{S}_N$.

Theorem 4 allows us to design a cyclic convolution algorithm for any length $N$ (not divisible by $p$) from an algorithm of length $p^n - 1$ provided $N | (p^n - 1)$. Though many values of $n$ may satisfy this condition, practical considerations suggest that we choose the smallest such $n$. Furthermore, when $N$ itself is of the form $p^n - 1$ as in Section III, $N' = 1$ and hence, $\mathbb{S}_N = \mathbb{S}$, $A' = A$, $B' = B$, and $C' = C$. Thus, the algorithm design of Section III is only a special case of the more general procedure outlined in this section.

## V. Examples

In this section we demonstrate the application of the design method developed in Sections III and IV to construct certain cyclic convolution algorithms.

### A. Length 15 Algorithm over the Field of Constants GF(2)

In this case, $S_0 = \{0\}$, $S_1 = \{1, 2, 4, 8\}$, $S_3 = \{3, 6, 12, 9\}$, $S_5 = \{5, 10\}$, $S_7 = \{7, 14, 13, 11\}$; thus, $\mathbb{S} = \{0, 1, 3, 5, 7\}$.

Let $\alpha$ denote the primitive element of GF$(2^4)$ satisfying $1 + \alpha + \alpha^4 = 0$. The only $\theta$'s of the form $\theta = (2^n - 1)/(2^t - 1) = 15/(2^t - 1)$ for some integer $t$ in this case are 1 (for $t = 4$) and 5 (for $t = 2$). When $\theta = 1$, $P_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) = x^4 + x + 1$. Since $\sigma_1 = 4$, using Algorithm D of Appendix II, we obtain the first four columns of $A_1$ as

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \\ 1 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 1 & \\ 0 & 0 & 1 & 1 & \\ 1 & 0 & 1 & 0 & \\ 0 & 1 & 0 & 1 & \\ 1 & 1 & 1 & 1 & \end{bmatrix}.$$

The nine rows of $A_1$ correspond to $m_0$ to $m_8$ of Algorithm D. To fill up the remaining portion of $A_1$, we extend each row of $A_1$ by the MLRS generated by $P_1(x)$, namely,

$$1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1, \quad 1 \quad 0 \quad 0 \quad 0 \quad \cdots.$$

We then obtain

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Sampling the rows of $A_1$ with a sampling period 3 and 7, in accord with Theorem 2, we have $A_3$ and $A_7$ as

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

and

$$A_7 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

When $\theta = 5$, $\sigma_5 = 2$ and Algorithm B of Appendix II can be used to obtain the first two columns of $A_5$ directly. The remaining portion of $A_5$ can be filled in by extending each row of $A_5$ by the MLRS generated by $P_5(x) = (x - \alpha^5)(x - \alpha^{10}) = x^2 + x + 1$, namely,

$$1 \quad 0 \quad 1, \quad 1 \quad 0 \quad 1, \quad \cdots.$$

Thus

$$A_5 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Finally, from the symmetry of Algorithms B and D (with respect to the $x$ and $y$ sequences), we have that the matrix $B$ equals the matrix $A$ where, $A = [A_0^T, A_1^T, A_3^T, A_7^T, A_5^T]^T$, $A_0 = [1\ 1\ \cdots\ 1]$.

To compute the $C$ matrix we must again begin by choosing $\theta = 1$. Since $\sigma_1 = 4$, (5) needs to be evaluated for $k = 0, 1, 2, 3$ to get the first four rows of $C_1$. The $r_m$'s in (5) refer to the coefficients of the product polynomial in Algorithm D of Appendix II, and can be expressed in terms of the multiplications $m_i$. We now have from (5),

$$w_1(k) = \frac{1}{15} \sum_{m=0}^{6} r_m \operatorname{tr}(\alpha^{m-k}).$$

The trace function can be shown to have the following values over GF$(2^4)$

$$\mathrm{tr}(\alpha^i) = \begin{cases} 1, & i = 3,6,7,9,11,12,13,14; \\ 0, & \text{otherwise.} \end{cases}$$

We also have $(1/15) = 1$ in GF(2). Using these results we obtain

$$w_1(0) = r_3 + r_6 = m_0 + m_1 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8,$$

$$w_1(1) = r_0 + r_4 = m_0 + m_1 + m_3 + m_4 + m_7,$$

$$w_1(2) = r_0 + r_1 + r_5 = m_1 + m_2 + m_3 + m_4 + m_5,$$

$$w_1(3) = r_0 + r_1 + r_2 + r_6 = m_0 + m_2 + m_3 + m_4 + m_6.$$

Thus, the first four rows of $C_1$ are

$$C_1 = \left[\begin{array}{cccc} 1&1&0&1 \\ 1&1&1&0 \\ 1&0&1&1 \\ 1&1&1&1 \\ 0&1&1&1 \\ 1&0&1&0 \\ 1&0&0&1 \\ 1&1&0&0 \\ 1&0&0&0 \end{array} \cdots \right]^T.$$

The remaining rows of $C_1$ are obtained by simply extending each column by the pseudorandom sequence generated by the reciprocal polynomial of $1 + x + x^4$, i.e., $1 + x^3 + x^4$. This sequence is

$$1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1,\ 1\ 1\ 0\ 1\ \cdots.$$

Thus

$$C_1 = \begin{bmatrix} 1&1&0&1&0&1&1&0&0&1&0&0&0&1&1 \\ 1&1&1&0&1&0&1&1&0&0&1&0&0&0&1 \\ 1&0&1&1&0&0&1&0&0&0&1&1&1&1&0 \\ 1&1&1&1&0&1&0&1&1&0&0&1&0&0&0 \\ 0&1&1&1&1&0&1&0&1&1&0&0&1&0&0 \\ 1&0&1&0&1&1&0&0&1&0&0&0&1&1&1 \\ 1&0&0&1&0&0&0&1&1&1&1&0&1&0&1 \\ 1&1&0&0&1&0&0&0&1&1&1&1&0&1&0 \\ 1&0&0&0&1&1&1&1&0&1&0&1&1&0&0 \end{bmatrix}^T.$$

Sampling the columns of $C_1$ with a period 3 or 7 in accord with (8) immediately provides $C_3$ and $C_7$ as

$$C_3 = \begin{bmatrix} 1&1&1&1&0&1&1&1&1&0&1&1&1&1&0 \\ 1&0&1&0&0&1&0&1&0&0&1&0&1&0&0 \\ 1&1&1&0&1&1&1&1&0&1&1&1&1&0&1 \\ 1&1&0&0&0&1&1&0&0&0&1&1&0&0&0 \\ 0&1&1&1&1&0&1&1&1&1&0&1&1&1&1 \\ 1&0&0&0&1&1&0&0&0&1&1&0&0&0&1 \\ 1&1&0&1&1&1&1&0&1&1&1&1&0&1&1 \\ 1&0&0&1&0&1&0&0&1&0&1&0&0&1&0 \\ 1&0&1&1&1&1&0&1&1&1&1&0&1&1&1 \end{bmatrix}^T$$

and

$$C_7 = \begin{bmatrix} 1&0&1&1&1&1&0&0&0&1&0&0&1&1&0 \\ 1&1&1&1&0&0&0&1&0&0&1&1&0&1&0 \\ 1&0&0&1&1&0&1&0&1&1&1&1&0&0&0 \\ 1&1&0&0&0&1&0&0&1&1&0&1&0&1&1 \\ 0&0&0&1&0&0&1&1&0&1&0&1&1&1&1 \\ 1&0&1&0&1&1&1&1&0&0&0&1&0&0&1 \\ 1&1&1&0&0&0&1&0&0&1&1&0&1&0&1 \\ 1&0&0&0&1&0&0&1&1&0&1&0&1&1&1 \\ 1&1&0&1&0&1&1&1&1&0&0&0&1&0&0 \end{bmatrix}^T.$$

Finally, when $\theta = 5$, $\sigma_5 = 2$ and (5) needs to be evaluated to get the first two rows of $C_5$. We now obtain from (5)

$$w_5(k) = \frac{1}{15} \sum_{m=0}^{2} r_m \,\mathrm{tr}(\alpha^{5m-5k}),$$

where tr denotes the function from GF$(2^2) \to$ GF(2) defined by $\mathrm{tr}(\beta) = \beta + \beta^2$, or

$$\mathrm{tr}(1) = 0, \quad \mathrm{tr}(\alpha^5) = \mathrm{tr}(\alpha^{10}) = 1.$$

Also note that the $r_m$'s in (5) now refer to the coefficients of the product polynomial in Algorithm B of Appendix II, and can be expressed in terms of the $m_i$'s; thus

$$w_5(0) = r_1 + r_2 = m_0 + m_2,$$

$$w_5(1) = r_0 + r_2 = m_0 + m_1.$$

These relations provide the first three rows of $C_5$. The remaining portion of $C_5$ is filled by completing each column using the MLRS generated by $P_{-5}(x)$, which is

$$1\ 0\ 1,\ 1\ 0\ 1,\ \cdots.$$

Proceeding in this manner, we obtain

$$C_5 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}^T.$$

Finally choosing $C_0 = [1\ 1\ \cdots\ 1]^T$ we obtain $C$ as $C = [C_0\ C_1\ C_3\ C_7\ C_5]$.

### B. Length $p-1$ Algorithm over a Field of Constants GF($p$)

In this case, for each $i$ $S_i = \{i\}$, $\sigma_i = 1$, and $\mathbb{S} = \{0, 1, 2, \cdots, p-2\}$. The matrices $A_1$ and $B_1$ are $1 \times (p-1)$ matrices whose first elements are 1, from Algorithm A of Appendix II, and the remaining elements are filled in by the MLRS over GF($p$) of period $(p-1)$ generated by $P_1(x)$. For other values of $i$, the $A_i$ or $B_i$ row vectors are obtained by sampling $A_1$ with a sampling period $i$. This gives the $A$ and $B$ matrices. The $C$ matrix is constructed in a similar fashion, except that $P_{-1}(x)$ is used in its construction. It is easy to show that the $(p-1) \times (p-1)$ $A$ and $B$ matrices so constructed are Fourier matrices and $C$, an inverse Fourier matrix. Thus, the conventional approach of computing the convolution through the product of Fourier transforms is a special case of the general technique presented here when the field of constants is expanded to contain the $N$th roots of unity.

### C. Length 5 Algorithm over a Field of Constants GF(2)

Since $5|2^4 - 1 = 15$, we may base the length 5 algorithm on the length 15 algorithm of Section V-A. Clearly $\mathbb{S}_5 = \{0, 3\}$ and, therefore, using the matrices $A_0, A_3, C_0, C_3$ of the length 15 algorithm we obtain the matrices of length 5 algorithm from Theorem 4 as

$$A = B = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

and

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

The three examples presented here illustrate a wide variety of algorithms that may be generated using the procedures of this paper. The computational complexity of these algorithms is investigated in the next section.

## VI. Computational Complexity

It is clear from Sections III and IV that the construction of the algorithm is based on the algorithms for multiplying two $\sigma_\theta - 1$ degree polynomials for all possible values of $\theta \in \mathbb{S}_N$. But P2) of Section II shows that when $N$ equals or divides $p^n - 1$, these $\sigma_\theta$'s are divisors of $n$. Thus the availability of good degree $t-1$ polynomial multiplication algorithms for all the divisors $t$ of $n$ is desirable for the application of the technique described. The multiplicative complexity of the new algorithm is also directly dependent upon the complexities of the polynomial multiplication algorithms.

TABLE I
Complexity of Cyclic Convolutional Algorithm Over a Field of Constants GF(2)

| Length $N$ | No. of Mults. $M(N)$ | $M(N)/N$ |
|---|---|---|
| 3 | 4 | 1.3333 |
| 7 | 13 | 1.8571 |
| 15 | 31 | 2.0667 |
| 63 | 178 | 2.8254 |
| 255 | 841 | 3.2980 |
| 511 | 2029 | 3.9707 |
| 4095 | 18295 | 4.4676 |

TABLE II
Complexity of Cyclic Convolutional Algorithm Over a Field of Constants GF(3)

| Length $N$ | No. of Mults. $M(N)$ | $M(N)/N$ |
|---|---|---|
| 2 | 2 | 1 |
| 8 | 11 | 1.375 |
| 26 | 50 | 1.9231 |
| 80 | 173 | 2.1625 |
| 728 | 2147 | 2.9492 |

TABLE III
Complexities of Cyclic Convolutions of some Factor Lengths over a Field of Constants GF(2)

| Length $N$ | No. of Mults. $M(N)$ | $M(N)/N$ |
|---|---|---|
| 5 | 10 | 2 |
| 9 | 22 | 2.4444 |
| 13 | 55 | 4.2308 |
| 17 | 55 | 3.2353 |
| 21 | 52 | 2.4762 |
| 35 | 130 | 3.7143 |
| 45 | 157 | 3.4889 |
| 51 | 166 | 3.2549 |
| 65 | 280 | 4.3077 |
| 73 | 289 | 3.9589 |
| 85 | 280 | 3.2941 |
| 117 | 508 | 4.3419 |
| 315 | 1285 | 4.0794 |

The field of constants is always a subfield of the field over which the data sequences are defined. We assume that the only time consuming operation is the product of elements of this larger field. Thus, since the matrices $A$, $B$, and $C$ contain elements from the field of constants, these matrices multiply with vectors without contributing much to the overall complexity. The total complexity of the algorithm therefore equals the total number of multiplications one encounters in the product $Au \times Bv$. From the number of rows in the matrices $A$ or $B$, we may express the total complexity, $M(N)$, of the length $N$ algorithm as

$$M(N) = \sum_{l \in \mathbb{S}_N} R(l),$$

where $R(l)$ denotes the multiplicative complexity of degree $l-1$ polynomial multiplication. Tables I through IV list the values of $M(N)$ for various lengths $N$ and field of constants GF(2) and GF(3).

TABLE IV
COMPLEXITIES OF CYCLIC CONVOLUTIONS OF SOME FACTOR LENGTHS OVER A FIELD OF CONSTANTS GF(3)

| Length $N$ | No. of Mults. $M(N)$ | $M(N)/N$ |
|---|---|---|
| 4 | 5 | 1.25 |
| 5 | 10 | 2.0 |
| 7 | 19 | 2.7143 |
| 10 | 20 | 2.0 |
| 13 | 25 | 1.9231 |
| 14 | 38 | 2.7143 |
| 16 | 29 | 1.8125 |
| 20 | 41 | 2.05 |
| 28 | 77 | 2.75 |
| 40 | 83 | 2.075 |
| 52 | 125 | 2.4039 |
| 56 | 155 | 2.7679 |
| 91 | 259 | 2.8462 |
| 104 | 275 | 2.6442 |
| 182 | 518 | 2.8462 |
| 362 | 1061 | 2.9148 |

The results of this paper including the technique to design the bilinear algorithm *remain valid* even if the prime $p$ is replaced everywhere by a power of $p$. This increases the field of constants and thereby reduces the number of multiplications, consistent with the results of [17]. Table V lists the complexities of some of the cyclic convolution algorithms based on GF(4). By comparing Tables I and V, we can see that if the data sequences are over GF($2^n$) ($n \geq 2$), then by enlarging the field of constants from GF(2) to GF(4) saves approximately 30 percent of the multiplications. The practical disadvantage of this enlargement of the field of constants is that the matrices $A$, $B$, and $C$ would now have elements from GF(4) rather than from GF(2) and therefore their products with vectors may be more time consuming.

The *computational superiority* of the results obtained here may be appreciated by comparing them with earlier results. Reed, Truong, Miller, and Benjauthrit [18] have obtained cyclic convolution algorithms for data sequences over GF($2^m$). Their algorithm of length 15 over the field of constants GF(2) uses 40 multiplications which is almost 29 percent more than our algorithm of the same length. The advantage is clearly due to the fact that our algorithm is designed for length 15 whereas theirs is obtained from algorithms of lengths 3 and 5. Another method of obtaining the algorithms over finite fields is to adapt the available algorithms over the field of rationals to suit the finite fields. For example, reducing every constant in the Agarwal–Cooley cyclic convolution algorithms [1] modulo 2, one may obtain algorithms with field of constants GF(2). Using this approach, a cyclic convolution over GF(2) of length 63, will thus be evaluated through convolutions of lengths 7 and 9, requiring a total of $19 \times 22 = 418$ multiplications, which is more than double the number of multiplications required in our algorithm. It is, however, true that because of the transformation of the constants, the algorithms adapted to finite fields might be computationally less complex than the original algorithms over the rationals. But, it is generally very hard to identify the redundant multiplications. Finally, it may be mentioned

TABLE V
COMPLEXITY OF CYCLIC CONVOLUTIONAL ALGORITHM OVER A FIELD OF CONSTANTS GF(4)

| Length $N$ | No. of Mults. $M(N)$ | $M(N)/N$ |
|---|---|---|
| 15 | 21 | 1.4 |
| 63 | 123 | 1.95234 |
| 255 | 561 | 2.2 |
| 4095 | 12201 | 2.9795 |

that the earlier methods rely on constructing the algorithms of small lengths using the Chinese remainder theorem and obtaining algorithms of large lengths by combining those of smaller lengths. Those methods can therefore be used only if the given length $N$ is factored into relatively prime factors such that algorithm for each factor is available. For example, when $N = 255 = 17 \cdot 4 \cdot 3$, these methods fail because an algorithm of length 17 is not available, whereas, our methods give this algorithm using only 3.3 multiplications per point.

The computational effort in computing $Au$, $Bv$, and $Cm$ (where $u$ and $v$ are the data vectors and $m$ is the vector $Au \times Bv$) has been neglected so far. However, it is also possible to reduce this as well, using the theorems developed in this paper, as the following few preliminary results show.

a) For every $i \in \mathcal{S}_N$, the rows of $A_i$ are MLRS's over the field of constants, GF($p$), based on the same primitive polynomial of degree $n$, where $n$ is the smallest integer such that $N | p^n - 1$ (Theorem 3). Since only $n$ MLRS's generated by a primitive polynomial of degree $n$ may be linearly independent over GF($p$), it follows that only $n$ rows of $A_i$ are linearly independent. Therefore to compute $A_i u$, it is sufficient to compute the product of these $n$ rows with $u$ and then obtain the remaining components of the product vector from these $n$ products. To illustrate this, consider matrix $A_1$ of Example 1 of Section V with $N = 15$. The only rows of this matrix which are linearly independent over GF(2) are rows 1, 2, 4, and 5. Denote the product $A_1 u$ by the vector $a_1$ with components $a_1(i)$, $i = 1, 2, \cdots, 9$. To compute the $a_1$ vector, we may first evaluate $a_1(1)$, $a_1(2)$, $a_1(4)$, and $a_1(5)$ and then obtain the remaining elements as $a_1(3) = a_1(1) + a_1(2)$, $a_1(6) = a_1(4) + a_1(5)$, $a_1(7) = a_1(1) + a_1(4)$, $a_1(8) = a_1(2) + a_1(5)$, and $a_1(9) = a_1(7) + a_1(8)$.

The initial computation of $a_1(1)$, $a_1(2)$, $a_1(4)$, and $a_1(5)$ is also straightforward. Since the corresponding rows of $A_1$ are MLRS's over GF(2) from the same polynomial, they have $[N/2] = 8$ 1's each. Rows 1 and 2 have matching 1's at exactly 4 positions; rows 1, 2, and 3 have matching 1's at exactly 2 positions; rows 3 and 4 have exactly 4 matching 1's, etc. These matching positions may be exploited to reduce the total number of additions.

b) If for some $i \in \mathcal{S}_N$, $i | N$, then all the rows of that $A_i$ are periodic with a period $N/i$ (Theorem 2). Thus to multiply $A_i$ and $u$, we may first add every $N/i$th component of $u$ and then multiply the compressed $u$ with the nonperiodic portion of $A_i$. Thus, for the matrix $A_3$ of Example 1 of Section V, 3 divides $N = 15$ and hence $A_3 u = \bar{A}_3 \bar{u}$, where $\bar{A}_3$ is the matrix of the first five columns

of $A_3$ and $\bar{u}$ is a vector defined by

$$\bar{u}(i) = u(i) + u(i+5) + u(i+10), \qquad i = 0, 1, \cdots, 4.$$

Remarks a) and b) also apply to the matrix $B$. To simplify the product $Cm$, we may use similar principles.

c) For any $i \in \mathbb{S}_N$, the columns of $C$ are MLRS's based on the same primitive polynomial. If $n$ is the smallest integer such that $N | p^n - 1$, then any row of $C_i$ can be expressed as a linear sum (over GF$(p)$) of at most $n$ immediately previous rows. The linear relation is the difference equation based on the minimal polynomial of $\alpha^{-i}$. To illustrate this, consider matrix $C_1$ of Example 1 of Section V. It is based on the minimal polynomial $1 + x^3 + x^4$, and hence it may be verified that any $k$th row of $C_1$ $(k > 4)$ is equal to the sum over GF(2) of the $(k-4)$th and $(k-1)$th rows of $C_1$. This means that the product $C_1 m_1$ (where $m_1$ is that portion of $m$ which multiplies $C_1$) can be computed by first evaluating only the first four components and then calculating the remaining components through one addition each.

d) If $i \in \mathbb{S}_N$ and $i | N$, then the rows of $C_i$ are periodic with period $N/i$. Hence the product $C_i m_i$ is also periodic with the same period.

Application of these principles generally results in a *smaller number of additions* in the algorithm. The algorithm for cyclic convolution of length 5 over GF(2) derived as Example 3 of Section V, for example, has an additive complexity of only 27.

## VII. Application to the Decoding of Reed–Solomon Codes

Decoding of Reed–Solomon codes over GF$(2^n)$ in an efficient manner calls for the computation of a discrete Fourier transform (DFT) of length $N' = 2^n - 1$ over the field GF$(2^n)$ [18]. The standard method of implementing the DFT through the fast Fourier transformation (FFT) may not, however, be suitable in this case because $N'$ may not be factorizable into a sufficiently large number of factors.[3]

In particular, if $N'$ is a prime, one cannot construct the required DFT algorithm by combining the algorithms of smaller lengths. The only efficient method of computing such DFT's of prime lengths $N'$ is by relating them to the cyclic convolution of length $N' - 1$ as pointed out by Rader [20].

Thus, if $N'$ is prime, the DFT, $X$, of the length $N'$ sequence, $x$, is obtained in GF$(2^n)$ as

$$X(0) = \sum_{i=0}^{N'-1} x(i),$$

$$X\left(g^{N'-1-j}\right) = x(0) + w(j), \qquad j = 0, 1, \cdots, N' - 2,$$

where $w$ is the cyclic convolution of the length $N' - 1$ sequences $u$ and $v$ defined as

$$u(i) = x\left(g^i \bmod N'\right)$$

and

$$v(i) = \alpha^{g^{N'-1-i}}, \qquad i = 0, 1, \cdots, N' - 2,$$

$\alpha$ being a primitive element of GF$(2^n)$ and $g$, the generator of the multiplicative group of integers $\{1, 2, \cdots, N' - 1\}$.

Note that the computation of $u$ from $x$, or $X$ from $w$ is only a permutation (and an addition in the second case) which can be predetermined and programmed. The computational effort involved thus lies in the evaluation of the cyclic convolution $w$ from $u$ and $v$. This cyclic convolution is of length $N' - 1 = 2^n - 2$ and can be performed as a two-dimensional convolution, since $2^n - 2$ factors into relatively prime factors as $2^n - 2 = 2N$ where $N = 2^{n-1} - 1$. The cyclic convolution algorithm of length 2 is given in Appendix III and the algorithms of lengths $N = 2^{n-1} - 1$ are derived in the main body of this work. The multiplicative complexity of the resultant DFT algorithm is summarized in Table VI.

TABLE VI
Complexity of the Discrete Fourier Transform of Prime Length $N' = 2^n - 1$ over GF$(2^n)$

| $n$ | Length $N'$ | Multiplicative Complexity $M$ | $M/N'$ |
|---|---|---|---|
| 2 | 3 | 3 | 1.0000 |
| 3 | 7 | 12 | 1.7143 |
| 5 | 31 | 93 | 3.0000 |
| 7 | 127 | 534 | 4.2047 |
| 13 | 8191 | 54 885 | 6.7006 |

A comparison of these results with those of Reed *et al.*, [18] shows that our methods, when applicable, produce algorithms of *lower multiplicative complexities*. In particular, for $n = 5$, (length 31 DFT over GF(32)), Reed *et al.*, require 120 multiplications ([18, table I]), whereas, our algorithm requires only 93.

## VIII. Conclusion

Most of the efficient convolution algorithms available in the literature are *too specialized* and are not applicable to a wide variety of lengths. In addition, the available algorithms have been designed with *specific fields* in mind. If the user prefers to work over a different field, the *adapted* algorithm may not be optimal from a computational complexity standpoint.

In this work, we have developed a *structured design approach* for obtaining convolution algorithms over finite fields. The algorithms obtained are bilinear in nature, a dense distribution of lengths is available and the computational complexity is dependent on the field of constants. The bilinearity permits the convolution algorithm to be accomplished via transformations by three rectangular matrices much in the manner of Agarwal and Cooley. However, the design method presented here allows the matrices to be determined in an easy systematic fashion without resort to symbolic manipulation or tedious calcula-

[3] When $N'$ is factored as $N' = r_1 \cdot r_2 \cdots r_k$, the FFT requires $M_1 = N'(r_1 + r_2 + \cdots + r_k - k)$ multiplications. If $r_k$ is further factored as $r_k = s_1 s_2$ then the FFT will require $M_2 = N'(r_1 + r_2 + \cdots + r_{k-1} + s_1 + s_2 - (k+1))$ multiplications. Since $M_1 - M_2 = N'(r_k - s_1 - s_2 + 1) = N'(s_1 s_2 - s_1 - s_2 + 1) > 0$, we have $M_2 < M_1$. Thus, the larger the number of factors, the more efficient the FFT is.

tions. Only those lengths divisible by the field characteristics are not available. This represents a much milder restriction on algorithm utility. Furthermore, the design itself takes into account the underlying field and, therefore, represents a more optimal approach.

The algorithm design uses certain notions from the traditional view of a convolution as a product of Fourier transforms as opposed to the more recent formulation in terms of the Cook–Toom algorithm and the multidimensional techniques. The new algorithm is based on a number of results which reveal that the bilinear transformation matrices may be partitioned into submatrices, each of which corresponds to a specific cyclotomic set (Theorem 1). It is also shown that only a few of these submatrices are "fundamental" and the rest may be determined from these (Theorem 2). In addition, each fundamental submatrix is intimately related to a maximal length recurrent sequence (MLRS) (Theorem 3). Thus, the bilinear transformation matrices are constructed by filling the rows and columns of the submatrices by MLRS's whose initial conditions are determined by small degree polynomial multiplication algorithms. The computational complexity of the resultant algorithm is determined by the complexity of the small polynomial multiplication algorithms involved, the underlying field, and the overall length. Examples of the procedure are presented for lengths 5 and 15 algorithms over GF$(2^m)$, and the length $p-1$ algorithm over GF$(p)$.

## Appendix I

*Proofs of Properties P1)–P5) of Section II*

With $N = p^n - 1$, consider the correspondence of the integer set $\{0, 1, 2, \cdots, N-1\}$ with the elements of GF$(p^n)$ as $i \leftrightarrow \alpha^i$, where $\alpha$ is a primitive element of GF$(p^n)$. It is obvious from the manner in which the sets $S_i$ are defined, nonzero $j_1, j_2 \in S_i$ yield conjugate $\alpha^{j_1}$ and $\alpha^{j_2}$ ([16, p. 101]). Thus $\sigma_i$ equals the number of conjugates of $\alpha^i$ and $i \in \mathbb{S}$ if and only if it is the smallest power of $\alpha$ amongst all the conjugates of $\alpha^i$.

For any integer $t|n$, GF$(p^t) \subset$ GF$(p^n)$ ([16, th. 4.418]) and the element $\alpha^\theta$ generates the multiplicative group of GF$(p^t)$ when $\theta = (p^n - 1)/(p^t - 1)$. Every nonzero element of GF$(p^t)$ is of the form $\alpha^{\theta l}$ for positive integer $l$'s. Since all the conjugates of $\alpha^\theta$ belong to GF$(p^t)$, it is clear that for any conjugate $\alpha^j$, $j > \theta$. Hence $\theta \in \mathbb{S}$. Moreover, $\alpha^\theta$ generates the multiplicative group of GF$(p^t)$ implying that it should have $t$ conjugate, i.e., $\sigma_\theta = t$, as stated in P2).

For any nonzero $i \in \mathbb{S}$, $\alpha^i$ must belong to some smallest GF$(p^t) \subseteq$ GF$(p^n)$, in the sense that $\alpha^i$ does not belong to any subfield of GF$(p^t)$. Then with $\theta$ defined as above, $\alpha^i = \alpha^{\theta l}$ or $i = \theta \cdot l$ for some integer $l$. Moreover, $\alpha^i$ does not belong to any subfield of GF$(p^t)$ implies that it has $t$ conjugates, i.e., $\sigma_i = t = \sigma_\theta$. This proves P3).

P4) and P5) follow directly from these arguments and P1) is obvious.

## Appendix II

*Multiplication of Polynomials of Small Degrees*

Algorithms B and C are taken from [19] and Algorithm D, is derived from Algorithm B by multidimensional techniques, as suggested in [19]. These algorithms are valid over any field.

Algorithm A Degree: 0, $f_0 \cdot g_0 = r_0$
Computation: direct 1 multiplication.

Algorithm B Degree: 1, $(f_0 + f_1 z) \cdot (g_0 + g_1 z) = r_0 + r_1 z + r_2 z^2$
Computation: 3 multiplications.
Let $m_0 = f_0 \cdot g_0$
$m_1 = f_1 \cdot g_1$
$m_2 = (f_0 + f_1) \cdot (g_0 + g_1)$,
then $r_0 = m_0$
$r_1 = -m_0 - m_1 + m_2$
$r_2 = m_1$.

Algorithm C Degree: 2, $(f_0 + f_1 z + f_2 z^2) \cdot (g_0 + g_1 z + g_2 z^2) = r_0 + r_1 z + r_2 z^2 + r_3 z^3 + r_4 z^4$
Computation: 6 multiplications.
Let $m_0 = f_0 \cdot g_0$
$m_1 = f_1 \cdot g_1$
$m_2 = f_2 \cdot g_2$
$m_3 = (f_0 + f_1) \cdot (g_0 + g_1)$
$m_4 = (f_1 + f_2) \cdot (g_1 + g_2)$
$m_5 = (f_2 + f_0) \cdot (g_2 + g_0)$,
then $r_0 = m_0$
$r_1 = -m_0 - m_1 + m_3$
$r_2 = -m_0 + m_1 - m_2 + m_5$
$r_3 = -m_1 - m_2 + m_4$
$r_4 = m_2$.

Algorithm D Degree: 3, $(f_0 + f_1 z + f_2 z^2 + f_3 z^3) \cdot (g_0 + g_1 z + g_2 z^2 + g_3 z^3) = r_0 + r_1 z + \cdots + r_6 z^6$
Computation: 9 multiplications.
Let $m_0 = f_0 \cdot g_0$
$m_1 = f_1 \cdot g_1$
$m_2 = (f_0 + f_1) \cdot (g_0 + g_1)$
$m_3 = f_2 \cdot g_2$
$m_4 = f_3 \cdot g_3$
$m_5 = (f_2 + f_3) \cdot (g_2 + g_3)$
$m_6 = (f_0 + f_2) \cdot (g_0 + g_2)$
$m_7 = (f_1 + f_3) \cdot (g_1 + g_3)$
$m_8 = (f_0 + f_1 + f_2 + f_3) \cdot (g_0 + g_1 + g_2 + g_3)$
then $r_0 = m_0$
$r_1 = -m_0 - m_1 + m_2$
$r_2 = -m_0 + m_1 - m_3 + m_6$
$r_3 = m_0 + m_1 - m_2 + m_3 + m_4 - m_5 - m_6 - m_7 + m_8$
$r_4 = -m_1 + m_3 - m_4 + m_7$
$r_5 = -m_3 - m_4 + m_5$
$r_6 = m_4$.

## Appendix III

*Cyclic Convolution of Length 2 over a Field of Characteristic 2*

$$w = u * v$$

Computation:
Let $m_0 = (u(0) + u(1)) \cdot v(0)$
$m_1 = u(0) \cdot (v(0) + v(1))$
$m_2 = u(1) \cdot (v(0) + v(1))$,
then $w(0) = m_0 + m_2$
$w(1) = m_0 + m_1$.

## References

[1] R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 392–410, Oct. 1977.

[2] ——, "New algorithms for digital convolution," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Hartford, CT, pp. 360–362, May 1977.
[3] S. Winograd, "On computing the discrete Fourier transform," *Math. Comput.*, vol. 32, pp. 175–199, Jan. 1978.
[4] R. C. Agarwal and C. S. Burrus, "Fast convolution using Fermat number transforms with applications to digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 87–99, Apr. 1974.
[5] ——, "Number theoretic transforms to implement fast convolution," *Proc. IEEE*, vol. 63, pp. 550–560, Apr. 1975.
[6] C. M. Rader, "Discrete convolutions via Mersenne transform," *IEEE Trans. Comput.*, vol. C-21, pp. 1269–1273, Dec. 1972.
[7] I. S. Reed and T. K. Truong, "The use of finite fields to compute convolutions," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 208–213, Mar. 1975.
[8] ——, "Complex integer convolutions over a direct sum of Galois fields," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 657–661, Nov. 1975.
[9] N. S. Reddy and V. U. Reddy, "On convolutional algorithms for small word length digital filtering applications," *IEE J. Electron., Circuits, Syst.*, vol. 3, pp. 253–256, Nov. 1979.
[10] ——, "Combination of complex rectangular transforms and F.N.T. to implement fast convolution," *Electron. Lett.*, vol. 15, pp. 716–717, 25 Oct. 1979.
[11] V. U. Reddy and N. S. Reddy, "Complex rectangular transforms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Washington, DC, pp. 518–521, Apr. 1979.
[12] H. J. Nussbaumer and P. Quandalle, "Computations of convolutions and discrete Fourier transforms by polynomial transforms," *IBM J. Res. Develop.*, vol. 22, pp. 134–144, Mar. 1978.
[13] H. J. Nussbaumer, "Complex convolutions via Fermat number transforms," *IBM J. Res. Develop.*, vol. 20, pp. 282–284, Dec. 1976.
[14] ——, "Fast polynomial transform algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 205–215, Apr. 1980.
[15] D. E. Knuth, "Seminumerical algorithms," in *The Art of Computer Programming*, vol. 2. Reading, MA: Addison-Wesley, 1971.
[16] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[17] S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants," *Math. Syst. Theory*, vol. 10, pp. 169–180, 1977.
[18] I. S. Reed, I. K. Truong, R. L. Miller, and B. Benjauthrit, "Further results on fast transforms for decoding Reed–Solomon codes over GF($2^m$) for $m = 4, 5, 6, 8$," in *Deep Space Network Progress Report 42-50*, pp. 132–154, Jet Propulsion Laboratory, Pasadena, CA, Jan. 1979.
[19] R. C. Agarwal and C. S. Burrus, "Fast one dimensional convolution by multidimensional techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 1–10, Feb. 1974.
[20] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," in *Proc. IEEE*, vol. 56, pp. 1107–1108, June 1968.

# Correspondence

## Group Codes for the $M$-Receiver Gaussian Broadcast Channel

CHARLES DOWNEY AND JOHN K. KARLOF, MEMBER, IEEE

***Abstract*—The $M$-receiver Gaussian broadcast channel is a communication system in which a single codeword is transmitted over $M$ distinct Gaussian channels and is received by $M$ receivers. The receivers have no contact with each other and the channels have different signal-to-noise ratios. The purpose of this correspondence is to define the concept of group code for the $M$-receiver Gaussian broadcast channel and study permutation codes as a special case. Such a code is generated by an initial vector $x$, a group $G$ of orthogonal $n$-by-$n$ matrices, and a sequence of subgroups of $G$.**

### I. Introduction

The $M$-receiver Gaussian broadcast channel is a model of a communication system in which a single codeword is transmitted over $M$ distinct Gaussian channels and is received by $M$ receivers. The receivers have no contact with each other and the channels have different signal-to-noise ratios. According to the signal-to-noise ratio of its channel, each receiver decodes a different amount of information from the received word.

In 1972, Cover [4] introduced the concept of broadcast channel coding theory. Bergmans [1] established the capacity region of the two-receiver Gaussian broadcast channel. In these two papers random coding techniques were used. Recently, Heegard, dePedro, and Wolf [5] defined permutation codes for the two-receiver Gaussian broadcast channel and Downey and Karlof [2] defined group codes for the same channel. Heegard *et al.* examined the goodness of their codes in terms of error probabilities, while Downey and Karlof used the minimum distance between codewords.

The purpose of this correspondence is to define the concept of group code for the $M$-receiver Gaussian broadcast channel and study permutation codes as a special case. Such a code is generated by an initial vector $x$, a group $G$ of orthogonal $n$-by-$n$ matrices, and a sequence of subgroups of $G$. The subgroups are used to partition the codewords into subsets, called clouds. For each channel we form a different set of clouds. The codewords in the same cloud represent the same message to that channel's receiver. We state conditions on the subgroups and the initial vector that are needed to generate good codes (in terms of minimum distances). We also find "optimal initial vectors" for certain sequences of subgroups of the groups used to generate variant II permutation group codes in [2], [5], and [6].

### II. $M$-Receiver Group Codes

The source in Fig. 1. contains a library of $s$ equally likely codewords one of which is transmitted every $T$ seconds over all $M$ channels. Each channel has a different signal-to-noise ratio with the lower channel numbers corresponding to the higher ratios. Receiver 1 decodes the received word as one of $s = t_1$ messages, receiver 2 decodes the received word as one of $t_2$ messages ($t_1 > t_2$), and receiver 3 decodes the received word as one of $t_3$ messages ($t_1 > t_2 > t_3$); in general, receiver $i$ decodes the received word as one of $t_i$ messages ($t_1 > t_2 > \cdots > t_i$). Therefore, according to the signal-to-noise ratio of the channel, each receiver extracts a different amount of information from the

Manuscript received November 6, 1980; revised June 9, 1982. This work was supported in part by a Grant from the University of Nebraska at Omaha.

The authors are with the Department of Mathematics and Computer Science, University of Nebraska at Omaha, Omaha, NB 68182.

0018-9448/83/0700-0595$01.00 