ON THE COMPLEXITY OF MULTIPLICATION IN FINITE FIELDS

by

A. Lempel*, G. Seroussi* and S. Winograd†

Technical Report #214
August 1981

* Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel.

† IBM - T. J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.
ON THE COMPLEXITY OF MULTIPLICATION IN FINITE FIELDS

A. Lempel*, G. Seroussi* and S. Winograd†

ABSTRACT

In this paper we study the bilinear complexity of multiplying two arbitrary elements from an n-th degree extension \( \Phi \) of a finite field \( F \), and the related problem of multiplying, over \( F \), two polynomials of degree \( n-1 \) with indeterminate coefficients. We derive a new linear lower bound, and we describe an algorithm leading to a quasi-linear upper bound.

* Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel.
† IBM - T.J. Watson Research Center, Yorktown Heights, N.Y. 10598, U.S.A.
1. INTRODUCTION

Let $F$ be a field and let $\mathcal{E}$ be an extension of degree $n$ of $F$. Given a basis $\mathcal{B} = (\omega_1, \omega_2, ..., \omega_n)$ of $\mathcal{E}$, viewed as a vector space over $F$, every element $\alpha \in \mathcal{E}$ has a unique representation as a column vector $\alpha = (a_1, a_2, ..., a_n)^t$ over $F$, i.e., $\alpha = a_1\omega_1 = \sum_{i=1}^{n} a_i\omega_i$ (\text{A}^t$ denotes the transpose of $A$).

Let $M_{i j k}, 1 \leq i, j, k \leq n$ be elements of $\mathcal{E}$ such that $(\omega_i, \omega_j)_{\mathcal{B}} = (M_{i j 1}, M_{i j 2}, ..., M_{i j n})^t$, and consider two arbitrary elements $x, y \in \mathcal{E}$ where $x = (x_1, x_2, ..., x_n)^t$ and $y = (y_1, y_2, ..., y_n)^t$. Then

\[
xy = \left( \sum_{i=1}^{n} x_i \omega_i \right) \left( \sum_{j=1}^{n} y_j \omega_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i y_j M_{i j k}^t \omega_k 
\]

Hence, the coordinate values of $(xy)_{\mathcal{B}}$ form a system of bilinear forms in $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$.

This paper deals with the multiplicative complexity of this system of bilinear forms (and related ones) when $F$ is a finite field.

Let $\varphi$ be a vector of $s$ bilinear forms over $F$ in the indeterminates $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$, and let $x$ and $y$ denote the column vectors $(x_1, x_2, ..., x_n)^t$ and $(y_1, y_2, ..., y_n)^t$ respectively. Then we can write $\varphi = x y$ where $X$ is an $s \times n$ matrix whose entries are $F$-linear forms in $x_1, x_2, ..., x_n$.

A bilinear algorithm of length (complexity) $t$ for computing $\varphi$ over $F$ takes the form

\[
\varphi = C m(x, y),
\]

where $C$ is an $s \times t$ matrix with entries from $F$, and $m(x, y)$ is a
column vector whose entries (the multiplications of the algorithm) are of the form

\[ m_i(x, y) = \xi_i(x) \xi_i^*(y), \quad 1 \leq i \leq t. \]

Here \( \xi_i(x) \) and \( \xi_i^*(y) \) are F-linear forms in \( x_1, \ldots, x_n \) and \( y_1, \ldots, y_n \), respectively.

The bilinear complexity of \( \varphi \), denoted by \( \mu(\varphi) \), is the length of the shortest bilinear algorithm computing \( \varphi \).

Let \( \Omega \) and \( \Lambda \) be bases of \( \phi \) over \( F \), and let \( V \) be the matrix over \( F \) such that \( \Lambda = \Omega V \). If

\[ (xy)_\Omega = Cm(x_\Omega, y_\Omega) \]

is a bilinear algorithm for computing \( (xy)_\Omega \), then

\[ (xy)_\Lambda = V^{-1}Cm(Vx_\Lambda, Vy_\Lambda) = Cm^*(x_\Lambda, y_\Lambda) \]

is a bilinear algorithm of the same length for computing \( (xy)_\Lambda \). Thus, the bilinear complexity of multiplication in \( \phi \) is independent of the basis chosen. In particular, if \( \Omega = (1, \alpha, \alpha^2, \ldots, \alpha^{n-1}) \) where \( \alpha \in \phi \) is a root of a monic irreducible polynomial \( P(u) \) of degree \( n \) over \( F \), then the system of bilinear forms defined by \( (xy)_\Omega \) is the same as the system defined by the modular polynomial multiplication

\[ r_p(u) = x(u) \cdot y(u) \mod P(u), \]

where \( x(u) = \sum_{i=1}^{n} x_i u^{i-1} \) and \( y(u) = \sum_{i=1}^{n} y_i u^{i-1} \).

---

1 In this paper \( a(x) \mod b(x) \) denotes the residue of least degree of \( a(x) \mod b(x) \), so that \( \deg(a(x) \mod b(x)) < \deg b(x) \).
We shall denote this system of bilinear forms by \( E_p = x_p y \), the underlying field \( F \) and the degree \( n \) of the irreducible polynomial \( P(u) \) being understood from the context. One way of computing \( r_p \) is by first computing the coefficients of the polynomial product \( r(u) = x(u) \cdot y(u) \) and then reducing the result modulo \( P(u) \). Denoting by \( r \) the vector formed by the coefficients of \( r(u) \), it is clear that the entries of \( r_p \) are linear combinations (over \( F \)) of the entries of \( r \) and, hence, \( \mu(r_p) \leq \mu(r) \). On the other hand, if \( P(u) \) is of degree \( 2n-1 \), we have

\[
x(u) \cdot y(u) = x(u) \cdot y(u) \mod P(u)
\]

and hence, \( \mu(r) \leq \mu(r_p) \).

The problem of computing \( r \) and \( r_p \) has been extensively studied in the literature ([1],[2],[3],[4]), and rather strong results have been obtained for the case where \( F \) is "large enough". The main known facts can be summarized as follows:

(i) ([1],[2],[3]). Over any field \( F \), \( \mu(r) \geq 2n-1 \) and \( \mu(r_p) \geq 2n-1 \). If \( |F| \geq 2n-2 \), these lower bounds are tight, i.e. \( \mu(r) = \mu(r_p) = 2n-1 \).

(ii) ([3]). In any bilinear algorithm of length \( 2n-1 \) for computing \( r \), either the \( 2n-1 \) multiplications are of the form

\[
m_i(x,y) = (b_i x(a_i)) \cdot (c_i y(a_i)) \quad (a_i,b_i,c_i \in F, \ a_i \neq a_j \text{ for } i \neq j, 1 \leq i, j \leq 2n-1) \) or \( 2n-2 \) multiplications are of this form and the remaining multiplication is of the form \( (b_{2n-1} x_n) \cdot (c_{2n-1} y_n) \).

Hence, at least \( 2n-2 \) distinct elements from \( F \) (namely, the \( a_i \)) are needed to construct a bilinear algorithm of length \( 2n-1 \) for computing \( r \), and if \( |F| < 2n-2 \) then \( \mu(r) \geq 2n \).

(iii) ([4]). Every bilinear algorithm of length \( 2n-1 \) for computing \( r_p \) is equivalent to one that first computes \( r \) and then reduces
the result modulo $P(u)$. It follows, as in (ii), that if $|F| < 2n-2$ then $\mu(r_p) \geq 2n$.

In contrast to the above results, very little is known about the complexity of $r$ or of $r_p$ when $F$ is a finite field and $n$ is arbitrarily large. In fact, it is even unknown whether $\mu(r_p)$ or $\mu(r)$ are bounded by $c \cdot n$ for some positive constant $c$ independent of $n$. (Since $\mu(r_p) \leq \mu(r) \leq \mu(r_p)^T$ when $\deg P(u) = 2n-1$, a linear upper bound for one of the problems induces a linear upper bound for the other, and vice versa.) For additional motivation, a positive answer to this question would lead to an important result in the area of error correcting codes. Assume $|F| = q$. Lempel and Winograd have shown in [5] that a bilinear algorithm of length $t(n)$ for $r$ (or for $r_p$) gives rise to a $q$-ary linear code $C_n$ of length $N = t(n)$, dimension $k = n$ and minimum distance $d \geq n$. If there exists an infinite sequence $\{n_i\}_i=1^\infty$ such that $t(n_i) \leq c n_i$ for some constant $c$ and for all $i$, then the infinite sequence of codes $\{C_i\}_i=1^\infty$ is a "good sequence" [6, p. 269] in the sense that both $d/N$ and $k/N$ are bounded away from zero as $N$ tends to infinity. Known examples of such good sequences are rather scarce (see [6, p. 306]).
2. STATEMENT OF MAIN RESULTS

In this paper, we derive new lower and upper bounds for the bilinear complexity of \( r_p \) and of \( \mathcal{C} \) over finite fields.

In Section 3 we prove the following theorem:

**Theorem 1:** Over \( F = GF(q) \),

\[
\mu(r_p) \geq (2 + \frac{1}{q-1})n - ([\log_q an] + \log_q e),
\]

where \( a = \frac{q}{(q-1)\log_q e} \) and \( e = 2.71828... \).

This lower bound improves the previously known lower bound of \( 2n \) and, as expected, it is asymptotically \( 2n + o(n) \) as \( q \) increases. When \( q = 2 \) (i.e. \( F \) is the binary field \( GF(2) \)), the lower bound is maximal and it is asymptotically \( 3n + o(n) \).

In Section 4 we describe the construction of a bilinear algorithm for \( \mathcal{C} \) over \( GF(q) \) which, in a sense, is a generalization of the construction leading to minimal algorithms (of length \( 2n-1 \)) over infinite fields. The analysis of the complexity of this algorithm yields the following upper bound for \( \mu(\mathcal{C}) \) (and, hence, also for \( \mu(r_p) \)):

**Theorem 2:** Over \( F = GF(q) \),

\[
\mu(\mathcal{C}) \leq f_q(n) \cdot n,
\]

where \( f_q(n) \) is a very slowly growing function defined recursively by

\[
f_q(n) = 2f_q([\log_q 2(q-1)n]), \quad n \geq 4, \quad q \geq 2.
\]

For \( n < 4 \), \( f_q(n) \) is defined as follows:
\[
\begin{array}{ll}
1 & n = 1, \ q \geq 2 \\
3/2 & n = 2, \ q \geq 2 \\
5/3 & n = 3, \ q \geq 4 \\
2 & n = 3, \ 2 \leq q \leq 3 \\
\end{array}
\]

Since \( \lceil \log_q (q-1)n \rceil < n \) for \( q \geq 2 \) and \( n \geq 4 \), \( f_q(n) \) is well-defined. Also, it can be readily shown that \( f_q(n) \) is monotonically non-decreasing, and that \( f_q(n) \) is unbounded as \( n \) tends to infinity. However, the growth of \( f_q(n) \) is extremely slow; for example, \( f_2(n) \leq 32 \) for \( n \leq 2^{127} \), and \( f_5(n) \leq 20 \) for \( n \leq 10^{2.66 \times 10^9} \).

In fact, the asymptotic behavior of \( f_q(n) \) is similar to the behavior of the function \( \log \log \log \ldots \log n \), where \( \log_k n \) is the inverse of the function \( G(n) = q^q \ldots q \) height \( n \).

In particular, it can be shown that, asymptotically

\[
f_q(n) \leq \log \log \log \ldots \log n;
\]

\[
q \underbrace{q \cdots q}_{k \text{ times}}
\]

for any positive integer \( k \). For these reasons, we call the result of Theorem 2 a "quasi-linear" upper bound.
3. A LOWER BOUND

The following theorem from [1] will be used in the proof of Theorem 1:

**Theorem 3:** Let \( \varphi = X \psi \) be a system of bilinear forms over \( F \). If \( X \) has at least \( s \) columns which are linearly independent (over \( F \)) then \( \mu(\varphi) \geq s \).

For any vector \( u \) over \( F \), we denote by \( \text{wt}(u) \) the number of non-zero entries of \( u \).

**Lemma 1:** Let \( V \) be a \( k \times r \) matrix over \( F = GF(q) \). Then

\[
\sum_{\mathbf{v} \in F^k} \text{wt}(\mathbf{v}^\top V) \leq r(q-1)q^{k-1},
\]

with equality holding iff \( V \) does not contain an all-zero column.

**Proof:** Let \( r' \) denote the number of non-zero columns of \( V \), and let \( V_i \) denote the \( i \)-th column of \( V \). If \( V_i \neq 0 \), the set \( \{ \mathbf{v} \in F^k | \mathbf{v}^\top V_i = 0 \} \) is a sub-space of dimension \( k-1 \) of \( F^k \) and, hence, its cardinality is \( q^{k-1} \). If \( V_i = 0 \), the cardinality of the set is \( q^k \). Now, in the \( q^{k} \times r \) matrix formed by all the rows of the form \( \mathbf{v}^\top V, \mathbf{v} \in F^k \), each column corresponding to a non-zero column of \( V \) has \( q^k - q^{k-1} \) non-zero entries, and in the remaining columns all the entries are zero. Hence,

\[
\sum_{\mathbf{v} \in F^k} \text{wt}(\mathbf{v}^\top V) = r'(q-1)q^{k-1} \leq r(q-1)q^{k-1},
\]

with equality holding 'iff \( r = r' \).
Lemma 2: Let $r_p = x_p y$. Then for every vector $w \in F^n - \{0\}$, all the entries of $w' x_p$ are linearly independent (as linear forms in $x_1, \ldots, x_n$) over $F$.

Proof: Assume, contrary to the claim, that for some $v, w \in F^n - \{0\}$, $w' x_p v = 0$. Viewing $v$ as the vectorial representation of an element $v \in \Phi$, the vector $x_p v$ represents the product $x \cdot v$ in $\Phi$. If $x$ runs through all the elements of $\Phi$ then, since $v \neq 0$, $x_p v$ runs through all the vectors in $F^n$. Hence, since $w' x_p v = 0$ for any value of $x_1, \ldots, x_n$, we have $w' s = 0$ for all $s \in F^n$ which implies $w = 0$, a contradiction.

Q.E.D.

Notice that Lemma 2 implies, in particular, that the rows of $X_p$ are linearly independent.

Proof of Theorem 1: Let $r_p = Cm(x, y)$ be a bilinear algorithm of length $t$ for computing $r_p = x_p y$. Since the rows of $X_p$ are linearly independent, so are the rows of $C$ and, hence $C$ has $n$ linearly independent columns. We may assume without loss of generality that the first $n$ columns of $C$ are linearly independent (otherwise we permute the columns of $C$ and the entries of $m(x, y)$ accordingly). Therefore, there exists a nonsingular matrix $W$ over $F$ such that $WC = [I_n A]$, where $I_n$ is the identity matrix of order $n$ and $A$ is an $n \times (t-n)$ matrix over $F$. Let $\tilde{W}$ denote the matrix consisting of the first $k$, $1 \leq k \leq n$, rows of $W$. Then

$$\tilde{W} x_p y = \tilde{W} C m(x, y) = [I_k \vdots 0 \vdots \tilde{A}] m(x, y)$$

where $\tilde{A}$ denotes the first $k$ rows of $A$. 
Let \( V = [I_k \ A] \) and let \( \tilde{m}(x,y) \) be the vector obtained by deleting the entries \( m_{k+1}, m_{k+2}, \ldots, m_n \) from \( m(x,y) \). Then
\[
\tilde{w}x_p = \tilde{v}m(x,y).
\]

Clearly, the rows of \( \tilde{W} \) are linearly independent and therefore, for every column vector \( v \in F^k - \{0\} \) we have \( v^t \tilde{W} \neq 0 \). By Lemma 2, it follows that the \( n \) columns (entries) of \( v^t \tilde{w}x_p \) are linearly independent over \( F \) and, by Theorem 3, \( \mu(v^t \tilde{w}x_p) \geq n \). But
\[
\mu(v^t \tilde{w}x_p) = \tilde{v}^t \tilde{m}(x,y).
\]
is a bi-linear algorithm for \( v^t \tilde{w}x_p \) and clearly, its length can be reduced to \( \mu(v^t V) \). Hence, we must have
\[
\mu(v^t V) \geq \mu(v^t \tilde{w}x_p) \geq n, \ v \in F^k - \{0\}.
\]

Summing over all vectors \( v \) in \( F^k \) we obtain
\[
\sum_{v \in F^k} \mu(v^t V) \geq (q^k - 1)n.
\]
(The contribution of the zero vector is zero.) On the other hand, observing that \( V \) has \( t-n+k \) columns and \( k \) rows, we have, by Lemma 1,
\[
\sum_{v \in F^k} \mu(v^t V) \leq (t-n+k)(q-1)q^{k-1}.
\]

Combining the above two inequalities, we obtain
\[
(t-n+k)(q-1)q^{k-1} \geq (q^k - 1)n.
\]
Solving for \( t \) and rearranging terms, we have
\[
t \geq \left( 2 + \frac{1}{q-1} \right) n - \left( k + \frac{q}{q-1} nq^{-k} \right).
\]
The right-hand side of this inequality is maximum when $k = \lfloor \log_q a_n \rfloor$, $a = \frac{q}{(q-1)\log_q e}$; for this value of $k$, we obtain

$$t \geq \left(2 + \frac{1}{q-1}\right)n - \left(\lfloor \log_q a_n \rfloor + \log_q e\right).$$

Q.E.D.
4. A QUASI-LINEAR UPPER BOUND

4.1 Application of the Chinese Remainder Theorem

Let \( z(u) = \sum_{i=0}^{m-1} z_i u^i \) be a polynomial of degree \( m-1 \) with indeterminate coefficients, and let \( \mathbf{z} = (z_0, z_1, \ldots, z_{m-1})^t \) be the vector of coefficients of \( z(u) \). Let \( Q_i(u), 1 \leq i \leq h, \) be constant polynomials with coefficients in \( F \), such that \( \gcd(Q_i, Q_j) = 1 \) for \( i \neq j \), and \( \deg(\prod Q_i(u)) \geq m \), and let \( \mathbf{r}(i) \) be the (column) vector of coefficients of: \( r(i)_j(u) = z(u) \mod Q_i(u) \). The following is one version of the Chinese Remainder Theorem, the proof of which can be found elsewhere.

Theorem 4: There exists a matrix \( \mathbf{T} \) over \( F \) such that

\[
\mathbf{z} = \mathbf{T} \mathbf{r}
\]

The Chinese Remainder Theorem can be used to construct algorithms for \( r \) from algorithms for smaller problems as follows:

Let \( z(u) = r(u) = x(u) \cdot y(u) = \left( \sum_{i=1}^{n} x_i u^{i-1} \right) \left( \sum_{i=1}^{n} y_i u^{i-1} \right) \), and choose \( Q(u) = \prod Q_i(u) \) such that \( \deg(Q(u)) \geq 2n-1 \). If \( r(i) = c(i)_m(i)(x, y) \), \( 1 \leq i \leq h, \) is a bilinear algorithm of length \( t_i \) for \( r(i)_j(u) = x(u) \cdot y(u) \mod Q_i(u) \), then
is a bilinear algorithm of length \( t = \sum_{j=1}^{h} t_j \) for \( r \).

When \( |F| \geq 2n-2 \), we can use one of the following constructions to obtain algorithms of length \( 2n-1 \) for \( r \):

(A1) Choose \( 2n-1 \) distinct elements \( a_1, a_2, \ldots, a_{2n-1} \) of \( F \), and let

\[
Q(u) = \prod_{i=1}^{2n-1} (u-a_i).
\]

The computation of \( r^{(1)}(u) = x(u) \cdot y(u) \mod (u-a_i) \) takes one multiplication, namely \( x(a_i) \cdot y(a_i) \). Hence, \( x(u) \cdot y(u) \) is computed using \( 2n-1 \) multiplications. This is essentially the algorithm described in [1].

(A2) Choose \( 2n-2 \) distinct elements \( a_1, a_2, \ldots, a_{2n-2} \) of \( F \). The following identity can be readily verified:

\[
x(u) \cdot y(u) = x(u) \cdot y(u) \mod \prod_{i=1}^{2n-2} (u-a_i) + \prod_{i=1}^{2n-2} (u-a_i) \cdot x_{n-1} y_{n-1}.
\]

As in (A1), \( x(u) \cdot y(u) \mod \prod_{i=1}^{2n-2} (u-a_i) \) is computed by the Chinese Remainder Theorem using \( 2n-2 \) multiplications, and the \((2n-1)\)st multiplication is \( x_{n-1} y_{n-1} \).

It was shown in [3] that any algorithm of length \( 2n-1 \) for computing \( r \) uses one of the above constructions. It is clear that none of them can be used if \( |F| < 2n-2 \). We now describe a generalization of the construction (A1) that is valid in finite fields for any value of \( n \).
4.2 The Algorithm

For any positive integer \( k \), let \( \ell(k) \) denote the number of monic irreducible polynomials of degree \( k \) over \( F = \text{GF}(q) \). Let \( s(n) \) be the least integer such that \( \sum_{k=1}^{s(n)} \ell(k) \geq 2n-1 \), and let \( R_k \), \( 1 \leq k \leq s(n) \), be integers such that \( 0 \leq R_k \leq \ell(k) \) and \( \sum_{k=1}^{s(n)} kR_k = 2n-1 \) (the existence of such integers follows from the definition of \( s(n) \); see the Appendix). Let \( Q_k^j(u) \) denote the \( j \)-th irreducible polynomial of degree \( k \) over \( F = \text{GF}(q) \) (according to some ordering), and let

\[
Q(u) = \prod_{k=1}^{s(n)} \prod_{j=1}^{R_k} Q_k^j(u).
\]

It is clear from the above definitions that \( \deg(Q(u)) = 2n-1 \) and that the polynomials \( Q_k^j(u) \) are pairwise relatively prime. Hence, the polynomial \( r(u) = x(u) \cdot y(u) \) can be reconstructed from the residues

\[
r(kj)(u) = x(u) \cdot y(u) \mod Q_k^j(u), \quad 1 \leq k \leq s(n), \quad 1 \leq j \leq R_k.
\]

The following recursive procedure uses this fact to construct a bilinear algorithm for \( r \). We shall denote the length of this bilinear algorithm by \( t(n) \).

1. If \( 1 \leq n \leq 3 \) use any optimal algorithm for \( r \).

   Else, do steps 2, 3, 4 for \( 1 \leq k \leq s(n) \), \( 1 \leq j \leq R_k \), and then do step 5.

2. Reduce \( x(u) \) and \( y(u) \) modulo \( Q_k^j(u) \).

3. Compute \( (x(u) \mod Q_k^j(u)) \cdot (y(u) \mod Q_k^j(u)) \) using (recursively!) the bilinear algorithm of length \( t(k) \) for multiplying polynomials of degree \( k-1 \). (Clearly, we need \( s(n) < n \) for \( n \geq 4 \). We shall prove this inequality in the sequel.)

4. Reduce the result of step 3 modulo \( Q_k^j(u) \).

Steps 2 and 4 do not require multiplications that are counted, and
the net result of steps 2, 3, 4 is a bilinear algorithm

\[ r^{(kj)} = c^{(kj)} m^{(kj)} (x, y), \quad 1 \leq k \leq s(n), \quad 1 \leq j \leq R_k, \]

of length \( t(k) \) for \( r^{(kj)}(u) = x(u) \cdot y(u) \mod Q_{kj}(u) \).

5. Apply the Chinese Remainder Theorem to construct an algorithm for \( r \) from the algorithms for \( r^{(kj)} \). The length of this algorithm is

\[ t(n) = \sum_{k=1}^{s(n)} R_k t(k), \quad n \geq 4. \]

The following lemmas will lead to an explicit upper bound for \( t(n) \).

**Lemma 3:** \( q^{k-1} < k \mu(k) q^k \).

**Proof:** These inequalities follow immediately from the formula [6, p. 115]

\[ l_k = \frac{1}{k} \sum_{d \mid k} q^{k/d} \mu(d), \]

where \( \mu(d) \) is the Möbius function, and \( d \) runs through all the divisors of \( k \).

**Lemma 4:** \( s(n) \leq \lceil \log_q 2(q-1)n \rceil \).

**Proof:** Let \( s'(n) = \lceil \log_q 2(q-1)n \rceil \). Then, using the lefthand inequality of Lemma 3, we have

\[ \sum_{k=1}^{s'(n)} k \mu(k) > \sum_{k=1}^{s'(n)} q^{k-1}, \]

or

\[ \sum_{k=1}^{s'(n)} k \mu(k) > 1 + \sum_{k=1}^{s'(n)} q^{k-1} = q^{s'(n)} - 1 \]

\[ \geq \frac{2(q-1)n-1}{q-1} = 2n - \frac{1}{q-1} > 2n-1. \]

From the definition of \( s(n) \), it follows that \( s(n) \leq s'(n) \). Q.E.D.
An immediate consequence of Lemma 4, is that for $q \geq 2$, $s(n) < n$ when $n \geq 4$. This condition is necessary to ensure the correctness of the recursive procedure. We are ready now to prove the upper bound on $\mu(r)$ as claimed in Theorem 2.

Proof of Theorem 2: Clearly, $\mu(r) \leq t(n)$. We shall prove that $t(n) \leq f_q(n) \cdot n$, by induction on $n$. For $n = 1, 2, 3$, $f_q(n)$ was defined so that $f_q(n) \cdot n$ is the optimal value of $\mu(r)$ in each case; hence $t(n) \leq f_q(n) \cdot n$ holds with equality for $1 \leq n \leq 3$. Assume now that $n \geq 4$ and that $t(k) \leq f_q(k) \cdot k$ for $1 \leq k < n$. Recalling that for $n \geq 4$, $t(n) = \sum_{k=1}^{s(n)} R_k t(k)$ and $s(n) < n$, we have

$$t(n) \leq \sum_{k=1}^{s(n)} f_q(k) \cdot k R_k.$$

Since $f_q(k)$ is a non-decreasing function of $k$, and $\sum_{k=1}^{s(n)} k R_k = 2n - 1$, using the result of Lemma 4, we obtain:

$$t(n) \leq f_q(s(n)) \sum_{k=1}^{s(n)} k R_k = f_q(s(n)) \cdot (2n - 1)$$

$$\leq 2f_q([\log_2 (q-1)n]) \cdot n = f_q(n) \cdot n.$$  

Q.E.D.

It should be noted that for the sake of obtaining a closed formula for the upper bound on $\mu(r)$, the bound on $t(n)$ derived in the proof of Theorem 2 is not as tight as could have been obtained. In the Appendix we present a table of exact values of $t(n)$ that were obtained by direct computation of the recursive formula for $t(n)$. These values are quite lower than the upper bound. However, the following theorem shows that the non-linearity of the upper bound is an intrinsic property of the algorithm rather than resulting from an inaccurate estimation.
Theorem 5: For any positive constant $c$, there exists an integer $N$ such that $t(n) \geq cn$ for all $n \geq N$.

Proof: We assume without loss of generality that $c$ is an integer, and we prove the theorem by induction on $c$. For $c=1$ and $c=2$, if $n \geq 1 + \left\lfloor \frac{q+1}{2} \right\rfloor$ then $q < 2n-2$ and, by the results of [3], we have $t(n) \geq 2n$. Assume now that $c > 2$ and that the theorem holds for all $c'$, $1 \leq c' < c$. We shall prove it for $c+1$. Let $N$ be the least integer such that $t(n) \geq cn$ for all $n \geq N$, let $s = N+4$, and let $S$ be the integer satisfying

$$2N - 1 \leq \sum_{k=1}^{s} k \cdot 1^{(k)} \leq 2N.$$

We claim that $t(n) \geq (c+1)n$ for all $n \geq S$. To prove this claim, consider an integer $n \geq S$. We have

$$t(n) = \sum_{k=1}^{s} R_k \cdot t(k) \geq \sum_{k=N}^{s} R_k \cdot t(k).$$

Using the inductive hypothesis, and the definition of the $R_k$, we obtain

$$t(n) \geq \sum_{k=N}^{s} c \cdot k \cdot R_k = c \left[ \sum_{k=1}^{N-1} k \cdot R_k - \sum_{k=1}^{s} k \cdot R_k \right] = c \left[ 2n-1 - \sum_{k=1}^{N-1} k \cdot R_k \right].$$

Since $k \cdot R_k \leq k \cdot 1^{(k)} \leq q^k$, we have

$$\sum_{k=1}^{N-1} k \cdot R_k \leq \sum_{k=1}^{N-1} q^k < \frac{q^{N-1}}{q-1} \times \frac{s-4}{q-1}.$$

By the definition of the integer $N$, we have $s(N) = s$ which, together with Lemma 4, yields

$$\sum_{k=1}^{N-1} k \cdot R_k \leq \frac{1}{q^{N-1}} \cdot \left[ \log_q 2(q-1)N \right]^{-1} \leq \frac{1}{q^{N-1}} \cdot \left[ 1 + \log_q 2(q-1)N \right]^{-1} \leq \frac{1}{q^{N-1}} \cdot \left[ 1 + \log_q 2(q-1)S \right]^{-1}.$$
or
\[ \sum_{k=1}^{N-1} kR_k < \frac{2N}{3} \leq \frac{2n}{3} \leq \frac{n}{4} \]
for \( q \geq 2 \). Hence,
\[ t(n) > c(2n-1 - \frac{n}{4}) = c(\frac{7}{4}n - 1) \geq (c+1)n \]
for \( c \geq 2 \) and all \( n \geq \bar{N} \geq 2 \).

Q.E.D.
APPENDIX

Table of exact values of $t(n)$ over $F = GF(2)$

We used the recursive formula $t(n) = \sum_{k=1}^{\min(s, s(n))} R_k t(k)$,

where $s(n) = \min\{\sum_{k=1}^{s(n)} l_2(k) \geq 2n-1\}$, and the $R_k$, $1 \leq k \leq s(n)$ are

defined as follows:

$$R_k = \begin{cases} 
\frac{s(n)-1}{2n-1 - \sum_{i=1}^{s(n)-1} l_2(i)} & \text{if } k = s(n) \\
2 - \frac{s(n)-1}{\sum_{i=1}^{s(n)} l_2(i) + s(n)R_1} - (2n-1) & \text{if } 1 \leq k < s(n) \\
l_2(k) & \text{otherwise}
\end{cases}$$

It can be readily verified that $\sum_{k=1}^{s(n)} R_k \neq 2n-1$.

The initial values of $t(n)$ are: $t(1) = 1$, $t(2) = 3$, $t(3) = 6$. We also
list the values of $s(n)$, $t(n)/n$ and $f_2(n)$ (for a comparison with the
upper bound $t(n) \leq f_2(n) n$).
<table>
<thead>
<tr>
<th>$n$</th>
<th>$t(n)$</th>
<th>$s(n)$</th>
<th>$t(n)/n$</th>
<th>$f_2(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>1.00</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-</td>
<td>1.50</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>-</td>
<td>2.00</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>3</td>
<td>2.75</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>4</td>
<td>3.20</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>4</td>
<td>3.67</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>27</td>
<td>4</td>
<td>3.86</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>33</td>
<td>4</td>
<td>4.13</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>4</td>
<td>4.22</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>44</td>
<td>4</td>
<td>4.40</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
<td>4</td>
<td>4.45</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>55</td>
<td>5</td>
<td>4.58</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>63</td>
<td>5</td>
<td>4.85</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>66</td>
<td>5</td>
<td>4.71</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>76</td>
<td>5</td>
<td>5.07</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>81</td>
<td>5</td>
<td>5.06</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>87</td>
<td>5</td>
<td>5.12</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>95</td>
<td>5</td>
<td>5.28</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>98</td>
<td>5</td>
<td>5.16</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>5</td>
<td>5.40</td>
<td>16</td>
</tr>
<tr>
<td>50</td>
<td>321</td>
<td>6</td>
<td>6.42</td>
<td>16</td>
</tr>
<tr>
<td>100</td>
<td>706</td>
<td>7</td>
<td>7.06</td>
<td>16</td>
</tr>
<tr>
<td>500</td>
<td>4,053</td>
<td>10</td>
<td>8.11</td>
<td>32</td>
</tr>
<tr>
<td>1,000</td>
<td>8,451</td>
<td>11</td>
<td>8.45</td>
<td>32</td>
</tr>
<tr>
<td>5,000</td>
<td>45,381</td>
<td>13</td>
<td>9.08</td>
<td>32</td>
</tr>
<tr>
<td>10,000</td>
<td>93,347</td>
<td>14</td>
<td>9.33</td>
<td>32</td>
</tr>
<tr>
<td>50,000</td>
<td>494,139</td>
<td>16</td>
<td>9.88</td>
<td>32</td>
</tr>
<tr>
<td>100,000</td>
<td>1,004,214</td>
<td>17</td>
<td>10.04</td>
<td>32</td>
</tr>
<tr>
<td>300,000</td>
<td>3,096,262</td>
<td>19</td>
<td>10.32</td>
<td>32</td>
</tr>
<tr>
<td>500,000</td>
<td>5,159,417</td>
<td>19</td>
<td>10.32</td>
<td>32</td>
</tr>
<tr>
<td>600,000</td>
<td>6,227,935</td>
<td>20</td>
<td>10.38</td>
<td>32</td>
</tr>
<tr>
<td>1,000,000</td>
<td>10,547,935</td>
<td>20</td>
<td>10.55</td>
<td>32</td>
</tr>
</tbody>
</table>
REFERENCES


