Radboud University Nijmegen

Faculty of Science

Karatsuba Algorithm for multiplication of Linearized Polynomials

Thesis BSc Computing Science

Supervisor:
Simona Samardziska

Author:
Niels Feij

Second reader:
Peter Schwabe

June 2023

Abstract

This paper explores multiplication of linearized polynomials in the field $\mathbf{F}_{q^m}$ as an alternative for matrix multiplications in the field $\mathbf{F}_q^{n\times n}$. this is done by means of an implementation of the Karatsuba Algorithm for linearized polynomials. A comparison based on time complexity is made against Schoolbook multiplication. This comparison shows the Karatsuba algorithm to be faster for linearized polynomials of degree $n > 19$ independent of q and for even lower degrees n when q gets larger. Finally, this research shows naive matrix multiplication in $\mathbf{F}_q^{n\times n}$ to yield the most efficient computation over Karatsuba linearized polynomials multiplications over F_{q^m}.

1 Introduction

The security of the majority of our digital assets and online activities relies on the strength of the underlying cryptographic techniques. Public-key cryptography, specifically RSA [8] and Elliptic Curve Cryptography [3], plays a crucial role in establishing a secure cryptographic infrastructure.

However, with the continuous advancements in quantum computing, the long-term security of this infrastructure, including (previously) encrypted information and digital signatures, is being compromised. Once a fully functional quantum computer becomes available, all the currently standardized and widely-used public-key algorithms can be vulnerable to attacks that can be carried out in polynomial time using a quantum computer.

In response to this imminent threat to our existing public-key infrastructure, the National Institute of Standards and Technology (NIST) initiated a process in 2016 to seek, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms [6]. The goal is to have a new replacement standard in place by 2024. These algorithms are commonly referred to as post-quantum or quantum-safe algorithms.

Some of these algorithms tend to make use of matrix-heavy computations, mainly multiplications. This enforces researchers to look for better and faster alternatives. In this paper this alternative comes in the form of multiplication of linearized polynomials. Although a linearized polynomial and a matrix are different mathematical structures, they can be used to represent the same elements, thereby deeming the comparison of computational complexity in their respective 'worlds' interesting. This paper defines, explores and compares the implementation of the Karatsuba Algorithm (KA) [2], designed for the multiplication of multi-digit numbers, for the multiplication of linearized polynomials. A multiplication with lower complexity in the 'world' of linearized polynomials (i.e. F_{q^m}) might result in reconsideration of matrices (i.e. F_q^m) being the industry standard 'world' of computation.

In order to simplify the problem we assume that the maximum degree of every two elements which are multiplied is identical.

The work is organized as follows. Firstly, the concepts of the KA and linearized polynomials will be made clear. Secondly, case studies for the KA for linearized polynomials of degree 1, 2 and 3 are performed. Next, the outcomes of these exploratory applications will be combined in a generalization, which will be elaborated, evaluated and compared to Schoolbook multiplication. Lastly, KA for linearized polynomials will be compared to naive matrix multiplication (MM).

1.1 Related work: Karatsuba Algorithm with polynomial multiplication

In 2012, S. Mishra and M. Pradhan [5] used polynomial multiplication in the classical KA to multiply two numbers. This lead to to a single and recursive algorithm which has better time performance over KA with regular multiplication.

1.2 Karatsuba Algorithm for polynomial multiplication

In 2016, A. Weimerskirch and C. Paar [9] generalized the classical KA for polynomial multiplication to (i) polynomials of arbitrary degree and (ii) recursive use. Their research provides tables that describe the best possible usage of the KA for polynomials up to a degree of 127. The results of the paper are especially useful for efficient implementations of computations over fixed-size fields like $\mathbf{F}_{p^m}$.

2 Preliminaries: Schoolbook for polynomials

The naive way to multiply two polynomials is often referred to as the schoolbook method. Let $A(x)$ and $B(x)$ be polynomials of degree-d with $n = d + 1$ coefficients:

$$A(x) = \sum_{i=0}^{d} a_i x^i, \quad B(x) = \sum_{i=0}^{d} b_i x^i$$

Then the product $C(x) = A(x)B(x)$ is defined as

$$\sum_{i=0}^{d} \sum_{j=0}^{d} a_i b_j x^{i+j} \tag{1}$$

With $A(x)$ and $B(x)$ being 'simple' polynomials, this multiplication takes n^2 multiplications and $(n-1)^2$ additions.

2.1 Karatsuba Algorithm

The KA [2] makes uses of the reoccurring factors in a multiplication. It has a single iteration variant as well as a recursive (divide-and-conquer) variant. The classic recursive KA simplifies the multiplication of two n-digit numbers into three multiplications of n/2-digit numbers by making use of reoccurring factors. By repeating this reduction process, it ensures a maximum of $n^{\log_2 3} \approx n^{1.58}$ single-digit multiplications. This makes it significantly faster than the traditional algorithm, which performs n^2 single-digit products.

As mentioned in section 1.2, KA also has an implementation for the multiplication of polynomials which is derived by simple algebraic transformations of the naive (Schoolbook) multiplication. Multiplying two polynomials using the single iteration KA results in a $\frac{1}{2}n^2 + \frac{1}{2}n$ multiplications, and $\frac{5}{2}n^2 - \frac{7}{2}n + 1$ additions [9].

2.1.1 Example Karatsuba Algorithm for polynomials

For sake of clarification, let us work out the simplest instance of the KA for polynomials. Let $A(x)$ and $B(x)$ be degree-1 polynomials:

$$A(x) = a_1x + a_0, B(x) = b_1x + b_0$$

Let there be three auxiliary variables D_0, D_1 and $D_{0,1}$ given by:

$$\begin{aligned} D_0 &= a_0b_0 \\ D_1 &= a_1b_1 \\ D_{0,1} &= (a_0 + a_1)(b_0 + b_1) \end{aligned}$$

Then the polynomial $C(x) = A(x)B(x)$ is computed by:

$$C(x) = D_1x^2 + (D_{0,1} - D_0 - D_1)x + D_0$$

Using this method, we need **3** multiplication and **4** additions, with the schoolbook method we needed **4** multiplication and **1** addition. Thus winning 1 multiplication at the cost of 3 additional additions.

2.2 Linearized polynomials

Linearized polynomials [7] are polynomials of the following form:

$$a = \sum_{k=0}^{t} a_k x^{q^k} = \sum_{k=0}^{t} a_k x^{[k]}, \quad a_k \in \mathbf{F}_{q^m}$$

Where q is a prime power, $\mathbf{F}_q$ is a finite field with q elements and $\mathbf{F}_{q^m}$ is a extension field over $\mathbf{F}_q$. The notation $[k]$ is used to denote q^i. We say linearized polynomial a has q-degree $\deg_q a = \max\{k \in \mathbf{N} : a_k \neq 0\}$.

Multiplication of two linearized polynomials $A(x)$ and $B(x)$ is defined as the composition

$$C(x) = A(B(x))$$

The following section derives another general equation to represent multiplication for linearized polynomials which shows similarity to the Schoolbook algorithm for all polynomials.

2.3 Schoolbook for linearized polynomials

Let $A(x)$ and $B(x)$ be linearized polynomials with q-degree 2:

$$\begin{aligned} A &= a_0x^{[0]} + a_1x^{[1]} + a_2x^{[2]} \\ B &= b_0x^{[0]} + b_1x^{[1]} + b_2x^{[2]} \end{aligned}$$

Then the composition will be of the form:

$$\begin{aligned}
A(B(x)) &= a_0\left(b_0x^{[0]}+b_1x^{[1]}+b_2x^{[2]}\right)^{[0]}+a_1\left(b_0x^{[0]}+b_1x^{[1]}+b_2x^{[2]}\right)^{[1]}\\
&\quad+a_2\left(b_0x^{[0]}+b_1x^{[1]}+b_2x^{[2]}\right)^{[2]}\\
&= a_0b_0^{[0]}x^{[0][0]}+a_0b_1^{[0]}x^{[1][0]}+a_0b_2^{[0]}x^{[2][0]}+a_1b_0^{[1]}x^{[0][1]}+a_1b_1^{[1]}x^{[1][1]}\\
&\quad+a_1b_2^{[1]}x^{[2][1]}+a_2b_0^{[2]}x^{[0][2]}+a_2b_1^{[2]}x^{[1][2]}+a_2b_2^{[2]}x^{[2][2]}\\
&= a_0b_0^{[0]}x^{[0]}+a_0b_1^{[0]}x^{[1]}+a_0b_2^{[0]}x^{[2]}+a_1b_0^{[1]}x^{[1]}+a_1b_1^{[1]}x^{[2]}\\
&\quad+a_1b_2^{[1]}x^{[3]}+a_2b_0^{[2]}x^{[2]}+a_2b_1^{[2]}x^{[3]}+a_2b_2^{[2]}x^{[4]}
\end{aligned}$$

Since:

$$\begin{aligned}
x^{[1][2]} &= x^{p^1\times p^2}\\
&= x^{p^{1+2}}\\
&= x^{p^3} = x^{[3]}
\end{aligned}$$

Further simplification gives the following equation:

$$\sum_{i=0}^{2}\left(\sum_{j=0}^{2}a_ib_j^{[i]}x^{[i+j]}\right) = \sum_{i=0}^{deg_q a}\sum_{j=0}^{deq_q b}a_ib_j^{[i]}x^{[i+j]}$$

Which, since linearized polynomials are a subset of all polynomials, logically has the same general form as equation (1) and can therefore be seen as the Schoolbook multiplication of two linearized polynomials. This computation takes the same amount of multiplications (n^2) and additions ($(n-1)^2$), however the exponents of the b-term require an additional n^2 exponentiation operations over the earlier stated Schoolbook complexity. All operation are done in $\mathbf{F}_{q^m}$.

2.4 Normal bases

Normal bases facilitate calculations in finite fields and can therefore be used to reduce the computational complexity. We shortly summarize important properties of normal bases in the following [1]. A basis $B=\{\beta_0,\beta_1,\ldots,\beta_{m-1}\}$ of $\mathbf{F}_{q^m}$ over $\mathbf{F}_q$ is a normal basis if $\beta_i=\beta^{[i]}$ for all i, where $\beta\in\mathbf{F}_{q^m}$ is called a normal element. As shown in [4], there is a normal basis for any finite extension field $\mathbf{F}_{q^m}$ over $\mathbf{F}_q$. If we represent elements of $\mathbf{F}_{q^m}$ in a normal basis over $\mathbf{F}_q$, the operation $a\to a^{[j]}$ where $a\in\mathbf{F}_{\mathbf{q^m}}$, an operation known as the Frobenius automorphism, can be accomplished in $O(1)$ operations over $\mathbf{F}_{q^m}$ as follows. Let $\mathbf{A}=[A_1,\ldots,A_m]^T\in\mathbf{F}_q^{m\times1}$ be the vector representation of $a\in\mathbf{F}_{q^m}$ in a normal basis. Then, for any j, the vector representation of $a^{[j]}$ is given by $[A_{m-j},A_{m-j+1},\ldots,A_0,A_1,\ldots,A_{m-j-1}]^T$, which is just a cyclic shift of the representation of a.

3 Karatsuba for linearized polynomials

In order to find the generalization of the algorithm for arbitrary degree linearized polynomials, this paper explores multiplication of linearized polynomials of q-degree 1, 2 and 3. We will encounter the transformation $a_0b_0^{[1]}\to a_0b_0^{[0]}$ and explore it. Finally, a concluding generalization is defined.

3.1 Karatsuba algorithm for q-degree 1 linearized polynomial (e.g. $a_1x^{q^1}+a_0x^{q^0}$)

Let there be two q-degree-1 linearized polynomials A(x) and B(x) given by:

$$A(x) = a_1x^{[1]} + a_0x^{[0]} \text{ and } B(x) = b_1x^{[1]} + b_0x^{[0]}$$

Then, following the definition of multiplication for linearized polynomials:

$$\sum_{i=0}^{deg_q a} \sum_{j=0}^{deq_q b} a_i b_j^{[i]} x^{[i+j]} \tag{2}$$

the product C(x) = A(x) B(x) can be determined in the following manner:

$$\begin{aligned} C(x) &= a_1b_1^{[1]}x^{[2]} + a_1b_0^{[1]}x^{[1]} + a_0b_1^{[0]}x^{[1]} + a_0b_0^{[0]}x^{[0]} \\ &= a_1b_1^{[1]}x^{[2]} + (a_1b_0^{[1]} + a_0b_1^{[0]})x^{[1]} + a_0b_0^{[0]}x^{[0]} \end{aligned}$$

The coefficient of $x^{[1]}$ in the above linearized polynomial can be written as:

$$(a_1b_0^{[1]} + a_0b_1^{[0]}) = ((a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) - a_0b_0^{[1]} - a_1b_1^{[0]})$$

Let there be three auxiliary variables D_0, D_1 and $D_{0,1}$ given by:

$$\begin{aligned} D_0 &= a_0b_0^{[1]} \\ D_1 &= a_1b_1^{[0]} \\ D_{0,1} &= (a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) \end{aligned}$$

Then the linearized polynomial $C'(x)$ be can be written as:

$$\begin{aligned} C'(x) &= D_1x^{[2]} + (D_{0,1} - D_0 - D_1)x^{[1]} + D_0x^{[0]} \\ &= a_1b_1^{[0]}x^{[2]} + ((a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) - a_0b_0^{[1]} - a_1b_1^{[0]})x^{[1]} + a_0b_0^{[1]}x^{[0]} \\ &= a_1b_1^{[0]}x^{[2]} + ((a_0b_0^{[1]} + a_0b_1^{[0]} + a_1b_0^{[1]} + a_1b_1^{[0]}) - a_0b_0^{[1]} - a_1b_1^{[0]})x^{[1]} + a_0b_0^{[1]}x^{[0]} \\ &= a_1b_1^{[0]}x^{[2]} + (a_1b_0^{[1]} + a_0b_1^{[0]})x^{[1]} + a_0b_0^{[1]}x^{[0]}S \end{aligned}$$

Which looks similar to, but is not, the result $C(x)$ that we are looking for.

$$\begin{aligned} C(x) &= a_1b_1^{[1]}x^{[2]} + (a_1b_0^{[1]} + a_0b_1^{[0]})x^{[1]} + a_0b_0^{[0]}x^{[0]} \\ C'(x) &= a_1b_1^{[0]}x^{[2]} + (a_1b_0^{[1]} + a_0b_1^{[0]})x^{[1]} + a_0b_0^{[1]}x^{[0]} \end{aligned}$$

In order to improve in terms of complexity, we need the following operations:

$$\begin{aligned} a_0b_0^{[1]} &\to a_0b_0^{[0]} \\ a_1b_1^{[0]} &\to a_1b_1^{[1]} \end{aligned}$$

to be computational light (e.g. not factoring $a_0b_0^{[1]}$, lowering the exponent of b and multiply).

3.1.1 Exploring $a_0 b_0^{[1]} \rightarrow a_0 b_0^{[0]}$

In order to find a possible relation between $a_0 b_0^{[1]}$ and $a_0 b_0^{[0]}$, let us work out the terms. We will use the normal basis representation of the individual elements, with the use of a normal basis $(\mathcal{B}, \mathcal{B}^q, ..., \mathcal{B}^{q^{m-1}})$. This results in the following two vector representations:

$$a_0 = (\alpha_1, \alpha_2, ..., \alpha_m)$$

$$b_0 = (\beta_1, \beta_2, ..., \beta_m)$$

Then for any j the vector representation of $a^{[j]}$ is given by:

$$a^{[j]} = (\alpha_{m-j}, \alpha_{m-j+1}, ..., \alpha_{m-j-1}) \tag{3}$$

The first operation is the exponentiation to get $b_0^{[0]}$ and $b_0^{[1]}$ from b_0. Following the Frobenius automorphism for normal bases, we get:

$$\begin{aligned} b_0^{[0]} &= (\beta_{m-0}, \beta_{m-0+1}, ..., \beta_{m-0-2}, \beta_{m-0-1}) \\ &= (\beta_m, \beta_1, ..., \beta_{m-2}, \beta_{m-1}) \end{aligned}$$

And:

$$\begin{aligned} b_0^{[1]} &= (\beta_{m-1}, \beta_{m-1+1}, ..., \beta_{m-1-2}, \beta_{m-1-1}) \\ &= (\beta_{m-1}, \beta_m, ..., \beta_{m-3}, \beta_{m-2}) \end{aligned}$$

Let us write out $a_0 b_0^{[0]}$ with simple algebraic operations for proving purposes:

$$\begin{aligned} a_0 b_0^{[0]} &= b_0^{[0]} a_0 \\ &= (\beta_m, \beta_1, ..., \beta_{m-2}, \beta_{m-1})(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m) \\ &= (\beta_m(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad \beta_1(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad ..., \\ &\qquad \beta_{m-2}(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad \beta_{m-1}(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m)) \end{aligned}$$

Let us now write out $a_0 b_0^{[1]}$ with simple algebraic operations for proving purposes:

$$\begin{aligned} a_0 b_0^{[1]} &= b_0^{[1]} a_0 \\ &= (\beta_{m-1}, \beta_m, ..., \beta_{m-3}, \beta_{m-2})(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m) \\ &= (\beta_{m-1}(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad \beta_m(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad ..., \\ &\qquad \beta_{m-3}(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m), \\ &\qquad \beta_{m-2}(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m)) \end{aligned}$$

For simplification purposes, let us now define γ to represent the basis of a_0:

$$\gamma = (\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m)$$

This now gives:

$$a_0 b_0^{[0]} = (\beta_m \gamma, \beta_1 \gamma, ..., \beta_{m-2}\gamma, \beta_{m-1}\gamma)$$

And:

$$a_0 b_0^{[1]} = (\beta_{m-1}\gamma, \beta_m \gamma, ..., \beta_{m-3}\gamma, \beta_{m-2}\gamma)$$

Which shows us the following equivalence:

$$a_0 b_0^{[0]} \to a_0 b_0^{[1]} \equiv a_0 b_0^{[0]} \gg a_0 b_0^{[1]}$$

as well as:

$$a_0 b_0^{[1]} \to a_0 b_0^{[0]} \equiv a_0 b_0^{[1]} \ll a_0 b_0^{[0]}$$

as well as:

$$a_0 b_0^{[0]} \to a_0 b_0^{[2]} \equiv a_0 b_0^{[0]} \gg^2 a_0 b_0^{[2]}$$

where the i in $\gg^i$ denotes the offset of the shift.

3.1.2 Combining

With the establishing of the triviality of the relation between $a_0 b_0^{[1]}$ and $a_0 b_0^{[0]}$, let us explore the KA once more. Let there be three auxiliary variables D_{0^1}, D_{1^0} and $D_{0,1}$ given by:

$$D_{0^1} = a_0 b_0^{[1]}$$
$$D_{1^0} = a_1 b_1^{[0]}$$
$$D_{0,1} = (a_0 + a_1)(b_0^{[1]} + b_1^{[0]})$$

As well as D_{0^0} and D_{1^1}:

$$D_{0^0} = a_0 b_0^{[0]}$$
$$D_{1^1} = a_1 b_1^{[1]}$$

Then the linearized polynomial $C(x)$ can be written as:

$$C(x) = D_{1^1} x^{[2]} + (D_{0,1} - D_0 - D_1) x^{[1]} + D_{0^0} x^{[0]}$$

To summarize the implementation: Let there be two q-degree-1 linearized polynomials A(x) and B(x) given by:

$$A(x) = a_1 x^{[1]} + a_0 x^{[0]} \text{ and } B(x) = b_1 x^{[1]} + b_0 x^{[0]}$$

We have the following variables:

$$a_0, a_1, b_0, b_1$$

Compute the following auxiliary variables:

$$b_0^{[1]} \qquad \text{by } b_0 \gg^2 \tag{4}$$

$$b_1^{[0]} \qquad \text{by } b_1 \gg \tag{5}$$

$$D_{0^1} \quad a_0 b_0^{[1]} \qquad \text{by } a_0 \cdot b_0^{[1]} \tag{6}$$

$$D_{1^0} \quad a_1 b_1^{[0]} \qquad \text{by } a_1 \cdot b_1^{[0]} \tag{7}$$

$$D_{0^0} \quad a_0 b_0^{[0]} \qquad \text{by } a_0 b_0^{[1]} \ll \tag{8}$$

$$D_{1^1} \quad a_1 b_1^{[1]} \qquad \text{by } a_1 b_1^{[0]} \gg \tag{9}$$

$$D_{0,1} \quad (a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) \qquad \text{by } (a_0 + a_1) \cdot (b_0^{[1]} + b_1^{[0]}) \tag{10}$$

3.1.3 Concluding Karatsuba algorithm for q-degree 1 linearized polynomials

By using the above approach for acquiring the variables, the computation of $C(x)$:

$$C(x) = D_{1^1} x^{[2]} + (D_{0,1} - D_{0^1} - D_{1^0}) x^{[1]} + D_{0^0} x^{[0]}$$

requires **4** shift operations, **3** multiplications and **4** additions.

3.2 Karatsuba algorithm for q-degree 2 linearized polynomial (e.g. $a_2 x^{q^2} + a_1 x^{q^1} + a_0 x^{q^0}$)

Let there be two q-degree 2 linearized polynomials $A(x)$ and $B(x)$ given by:

$$A(x) = a_2 x^{[2]} + a_1 x^{[1]} + a_0 x^{[0]}, \quad B(x) = b_2 x^{[2]} + b_1 x^{[1]} + b_0 x^{[0]}$$

Then, following the earlier given definition of multiplication for linearized polynomials (2), the product $C(x) = A(x)B(x)$ can be determined in the following manner:

$$\begin{aligned} C(x) &= a_2 b_2^{[2]} x^{[4]} + a_2 b_1^{[2]} x^{[3]} + a_2 b_0^{[2]} x^{[2]} \\ &\quad + a_1 b_2^{[1]} x^{[3]} + a_1 b_1^{[1]} x^{[2]} + a_1 b_0^{[1]} x^{[1]} \\ &\quad + a_0 b_2^{[0]} x^{[2]} + a_0 b_1^{[0]} x^{[1]} + a_0 b_0^{[0]} x^{[0]} \\ &= a_2 b_2^{[2]} x^{[4]} + (a_2 b_1^{[2]} + a_1 b_2^{[1]}) x^{[3]} + (a_2 b_0^{[2]} + a_1 b_1^{[1]} + a_0 b_2^{[0]}) x^{[2]} \\ &\quad + (a_1 b_0^{[1]} + a_0 b_1^{[0]}) x^{[1]} + a_0 b_0^{[0]} x^{[0]} \end{aligned}$$

The coefficients of $x^{[1]}$, (part of) $x^{[2]}$ and $x^{[3]}$ in the resulting polynomial $C(x)$ can be written as:

$$\begin{aligned} (a_1 b_0^{[1]} + a_0 b_1^{[0]}) &= ((a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) - a_0 b_0^{[1]} - a_1 b_1^{[0]}) \\ (a_2 b_0^{[2]} + a_0 b_2^{[0]}) &= ((a_0 + a_2)(b_0^{[2]} + b_2^{[0]}) - a_0 b_0^{[2]} - a_2 b_2^{[0]}) \\ (a_2 b_1^{[2]} + a_1 b_2^{[1]}) &= ((a_1 + a_2)(b_1^{[2]} + b_2^{[1]}) - a_1 b_1^{[2]} - a_2 b_2^{[1]}) \end{aligned}$$

Let there be the following auxiliary variables:

$$
\begin{aligned}
D_{0^0} &= a_0 b_0^{[0]}, D_{0^1} = a_0 b_0^{[1]}, D_{0^2} = a_0 b_0^{[2]} \\
D_{1^0} &= a_1 b_1^{[0]}, D_{1^1} = a_1 b_1^{[1]}, D_{1^2} = a_1 b_1^{[2]} \\
D_{2^0} &= a_2 b_2^{[0]}, D_{2^1} = a_2 b_2^{[1]}, D_{2^2} = a_2 b_2^{[2]} \\
D_{0,1} &= (a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) \\
D_{0,2} &= (a_0 + a_2)(b_0^{[2]} + b_2^{[0]}) \\
D_{1,2} &= (a_1 + a_2)(b_1^{[2]} + b_2^{[1]})
\end{aligned}
$$

Then:

$$
\begin{aligned}
C(x) = D_{2^2} x^{[4]} &+ (D_{1,2} - D_{1^2} - D_{2^1}) x^{[3]} + (D_{0,2} - D_{0^2} - D_{2^0} + D_{1^1}) x^{[2]} + \\
&(D_{0,1} - D_{0^1} - D_{1^0}) x^{[1]} + D_{0^0} x^{[0]}
\end{aligned}
$$

3.2.1 Combining

First, we need to pre-compute all individual b_i from $D_{0,1}$, $D_{0,2}$ and $D_{1,2}$, due to the earlier mentioned need for these single terms (section 3.1.3)

$$
b_0^{[1]}, b_0^{[2]}, b_1^{[0]}, b_1^{[2]}, b_2^{[0]}, b_2^{[1]}
$$

These can be computed by shifting b_i by an offset of 1, 2 or 3 in order to obtain $b_i^{[0]}$, $b_i^{[1]}$ and $b_i^{[2]}$ respectively. Coming down to a total of **6** shift operations.
With these b variables, $D_{0,1}$, $D_{0,2}$ and $D_{1,2}$ can be computed by means of **3** multiplications and **6** additions.
Next, we need one variation of D_{0^i}, D_{1^i} and D_{2^i}. For sake of nothing let us take the lowest possible exponent out of the above listing for b and compute the following auxiliary variables by means of **3** multiplications:

$$
\begin{aligned}
D_{0^1} &= a_0 \cdot b_0^{[1]} \\
D_{1^0} &= a_1 \cdot b_1^{[0]} \\
D_{2^0} &= a_2 \cdot b_2^{[0]}
\end{aligned}
$$

With the following use of shifts, all other variations of D_{0^i}, D_{1^i} and D_{2^i} can be computed.

$$
\begin{aligned}
D_{0^0} &= D_{0^1} \ll \\
D_{0^2} &= D_{0^1} \gg \\
D_{1^1} &= D_{1^0} \gg \\
D_{1^2} &= D_{1^0} \gg^2 \\
D_{2^1} &= D_{2^0} \gg \\
D_{2^2} &= D_{2^0} \gg^2
\end{aligned}
$$

3.2.2 Concluding Karatsuba algorithm for q-degree-2 linearized polynomials

Thus, with a total of **6** multiplications and **12** shifts, all auxiliary variables are computed and can be filled in in the following equation:

$$\begin{aligned}C(x) = D_{2^2}x^{[4]} + (D_{1,2} - D_{1^2} - D_{2^1})x^{[3]} + (D_{0,2} - D_{0^2} - D_{2^0} + D_{1^1})x^{[2]} + \\ (D_{0,1} - D_{0^1} - D_{1^0})x^{[1]} + D_{0^0}x^{[0]}\end{aligned}$$

Resulting in the product $C(x)$ by means of **13** addition operations.

3.3 Karatsuba algorithm for q-degree-3 linearized polynomial (e.g. $a_3x^{q^3} + a_2x^{q^2} + a_1x^{q^1} + a_0x^{q^0}$)

In order to make the generalization towards q-degree-n linearized polynomials, let us explore the KA for q-degree-3 linearized polynomials. Let there be two q-degree-3 linearized polynomials $A(x)$ and $B(x)$ given by:

$$\begin{aligned}A(x) = a_3x^{[3]} + a_2x^{[2]} + a_1x^{[1]} + a_0x^{[0]}, \\ B(x) = b_3x^{[3]} + b_2x^{[2]} + b_1x^{[1]} + b_0x^{[0]}\end{aligned}$$

The worked out execution of the KA can be found in Appendix A.

3.4 Generalization

With use of the insight created in the previous sections, a generalization of the KA for arbitrary q-degree linearized polynomials (e.g. $\sum_{i=0}^{n} a_i x^{q^i}$) is formed. Let there be two q-degree-n linearized polynomials (note: these polynomials have $n+1$ terms) $A(x)$ and $B(x)$ given by:

$$A(x) = \sum_{i=0}^{n} a_i x^{q^i} \text{ and } B(x) = \sum_{i=0}^{n} b_i x^{q^i} \tag{11}$$

Let $C(x)$ be the product of $A(x)$ and $B(x)$. The following auxiliary values are needed in order to perform the KA:

$$D_{i^j} = a_i b_i^{q^j} \quad [\forall i = 0, 1, 2, ..., n, \text{ and } \forall j = 0, 1, 2, ..., n]$$

$$\begin{aligned}D_{i,j} = (a_i + a_j)(b_{i^j} + b_{j^i}) \quad [\forall k = 1, 2, ..., 2(n+1) - 3, \\ \text{and } \forall i, j \in \{0, 1, ..., n\} \text{ such that } i + j = k \text{ and } j > i \geq 0]\end{aligned}$$

Then $C(x)$ of the form:

$$\sum_{i=0}^{2n} c_i x^{q^i}$$

Where:

$$\begin{aligned}c_0 = D_{0^0} \\ c_{2n} = D_{n^n}\end{aligned}$$

$$c_i = \begin{cases} \sum\limits_{\text{p+q=i,q>p}\geq 0} D_{p,q} - \sum\limits_{\text{p+q=i,q>p}\geq 0} (D_{p^q} + D_{q^p}), & \text{for odd values of i, } 0 < i < 2n \\ \sum\limits_{\text{p+q=i,q>p}\geq 0} D_{p,q} - \sum\limits_{\text{p+q=i,q>p}\geq 0} (D_{p^q} + D_{q^p}) + D_{i/2^{i/2}}, & \text{for even values of i, } 0 < i < 2n \end{cases} \tag{12}$$

3.4.1 Obtaining the auxiliary variables

When following the above algorithm by computing the auxiliary values blatantly, e.g. exponentiate without use of shifts, it would result in a complexity similar to the schoolbook approach. In order to profit from this algorithm, the earlier stated insights should be used for obtaining the auxiliary variables in such a way that computational cost is minimal. The following achieves just that:

$$b_{i^j} = b_i^{[j]} \quad [\forall i = 0,1,2,...,n, \text{ and } \forall j = 0,1,2,...,n \text{ such that } i \neq j] \tag{13}$$
$$D_{0^1} = a_0 b_{0^1} \tag{14}$$
$$D_{0^0} = D_{0^1} \ll \tag{15}$$
$$D_{0^i} = D_{0^1} \gg^i \quad [\forall i = 2,...,n] \tag{16}$$
$$D_{i^0} = a_i b_{i^0} \quad [\forall i = 1,2,...,n] \tag{17}$$
$$D_{i^j} = D_{i^0} \gg^j \quad [\forall i = 1,2,...,n \text{ and } \forall j = 1,2,...,n] \tag{18}$$
$$D_{i,j} = (a_i + a_j)(b_{i^j} + b_{j^i}) \quad [\forall k = 1,2,...,2(n+1)-3, \tag{19}$$
$$\text{and } \forall i,j \in \{0,1,...,n\} \text{ such that } i+j=k \text{ and } j > i \geq 0] \tag{20}$$

The set of exponentiations of b_i, referenced to as b_{i^j}, is obtained in step 13, this set has a size of $(n+1)^2 - (n+1) = n^2 + n$. These single term elements are part of the computation of the set D_{i^j} and are therefore needed. To compute all elements of b_{i^j}, $n^2 + n$ shift operations need to be performed.
Next, the variable D_{0^1} is computed instead of D_{0^0} because b_{0^1} is in b_{i^j} whereas b_{0^0} is not. This is the only element of D_{i^j} that is initially computed with $j = 1$ instead of $j = 0$. Thereby justifying step 15 and 16. All other D_{i^j} are computed by step 17 and 18. Combining step 14 - 18, there is a total of $n + 1$ multiplications (to obtain D_{0^1} and all D_{i^0}) and $n(n+1)$ shift operations.
Lastly, $D_{i,j}$ is computed. This set consists of $\frac{n^2+n}{2}$ elements and is computed with the use of tuples of elements of b_{i^j} by means of $\frac{n^2+n}{2}$ multiplications and $n^2 + n$ additions. All auxiliary variables are thus computed by means of:

$$(n^2 + n) + n(n+1) = n(2n+2) \qquad \text{exponentiations}$$
$$n + 1 + \frac{n^2+n}{2} = \frac{n^2+3n}{2} + 1 \qquad \text{multiplications}$$
$$n^2 + n \qquad \text{additions}$$

3.4.2 Concluding the generalization

In order to conclude the overall complexity, we need to determine the number of additions used in the computation of all c_i (excluding c_0 and c_{2n}). Let us denote that number by A.

Looking at this equation (12) together with the worked out example in section 3.3, we see the following. We require (at least) 2 additions for computing c_1, c_2, c_{2n-1} and c_{2n-2}, 5 additions for computing c_3, c_4, c_{2n-3} and c_{2n-4}. This increase of 3 additions goes on. In order to simplify the determination of the number of additions, let us consider the set of tuples:

$$\theta = \{(c_1, c_2, c_{2n-2}, c_{2n-1}), (c_3, c_4, c_{2n-4}, c_{2n-3}), \dots\}$$

Where $|\theta| = \frac{n}{2}$. Note that c_n is the only c_i that appears twice in this set. The elements of every tuple in θ share an equal number of additions in their computation (not yet taking the additional addition for even i into account). Thus, we can compute the following:

$$A' = 4\sum_{i=1}^{\frac{n}{2}}(3i-1) \tag{21}$$

However, the computation of c_n is counted twice by $i = \frac{n}{2}$ in the above equation, despite it only appearing once (see 3.3). The number of additions for c_n, we therefore have to subtract, is equal to:

$$A(c_n) = 3\frac{n}{2} - 1$$

And the number of additional additions due to the even i statement, we have to add, is equal to:

$$\frac{2n-2}{2} = n - 1$$

Combining gives A to be:

$$A = 4\sum_{i=1}^{\frac{n}{2}}(3i-1) - (3\frac{n}{2} - 1) + (n-1) = \frac{3n^2}{2} + \frac{n}{2} \tag{22}$$

Which lets us conclude the number of operations for the generalized KA for linearized polynomials of q-degree-n:

$$\begin{array}{cl} n(2n+2) & \text{exponentiations} \\ \frac{n^2+3n}{2} + 1 & \text{multiplications} \\ \frac{5n^2}{2} + \frac{3n}{2} & \text{additions} \end{array}$$

Table 1 below shows the number of operations for small prime n for both KA and Schoolbook multiplication. Note that with increase of n the number of multiplications needed for the KA gradually become less then for Schoolbook. Also note that the significant difference in number of exponentiations.

	KA			Schoolbook			
n	#MUL	#ADD	#EXP	#MUL	#ADD	#EXP	q^m
2	6	13	12	4	1	4	3^2
3	10	27	24	9	4	9	3^2
5	21	70	60	25	16	25	3^2
7	36	133	112	49	36	49	3^2
11	78	319	264	121	100	121	3^2
13	105	442	364	169	144	169	3^2

Table 1: comparison for KA and Schoolbook for small primes

4 Karatsuba vs Schoolbook

Next, we consider the complexity of the newly found implementation of the KA for linearized polynomials and compare that to the complexity of the Schoolbook multiplication. We define the operations to have the following complexities:

$$\begin{array}{rlr} \text{addition} & \text{in } \mathbf{F}_{q^m} & \mathcal{O}(m(\log_2(q)) \\ \text{multiplication} & \text{in } \mathbf{F}_{q^m} & \mathcal{O}(m\log_2(q)\log_2(m\log_2(q))) \\ \text{exponentiation} & \text{in } \mathbf{F}_{q^m} & \mathcal{O}(1) \end{array}$$

Note that the complexity of the exponentiation is constant since we assume the elements to be in their normal base representation. These complexities in combination with the number of executions established in 3.4.1 gives the following complexity for the **KA** for q-degree-n linearized polynomials:

$$\begin{aligned} \text{additions:} & \\ & (\frac{5n^2}{2} + \frac{3n}{2} - 1) \cdot \mathcal{O}(m(\log_2(q)) \\ & = \mathcal{O}((\frac{5n^2}{2} + \frac{3n}{2} - 1)m(\log_2(q)) \\ \text{multiplication:} & \\ & \frac{n^2+3n}{2} + 1 \cdot \mathcal{O}(\log_2(q)\log_2(m\log_2(q))) \\ & = \mathcal{O}(\frac{n^2+3n}{2} + 1(m\log_2(q)\log_2(m\log_2(q)))) \\ \text{shift:} & \\ & n(2n+2) \cdot \mathcal{O}(1) \\ & = \mathcal{O}(2n^2+2n) \end{aligned}$$

Combining into an overall complexity for the KA for q-degree-n linearized polynomials:

$$\begin{aligned} \mathcal{O}((&\frac{5n^2}{2} + \frac{3n}{2} - 1)m(\log_2(q)) + \\ &(\frac{n^2+3n}{2} + 1)(m\log_2(q)\log_2(m\log_2(q))) + 2n^2 + 2n) \end{aligned} \tag{23}$$

And for **Schoolbook**:

$$\begin{aligned}
\text{additions:} & \\
& (n-1)^2 \cdot \mathcal{O}(m(\log_2(q)) \\
&= \mathcal{O}((n-1)^2 m(\log_2(q)) \\
\text{multiplication:} & \\
& n^2 \cdot \mathcal{O}(\log_2(q)\log_2(m\log_2(q))) \\
&= \mathcal{O}(n^2(m\log_2(q)\log_2(m\log_2(q)))) \\
\text{Exponentiations:} & \\
& n^2 \cdot \mathcal{O}(1) \\
&= \mathcal{O}(n^2)
\end{aligned}$$

Combining into an overall complexity for Schoolbook for q-degree-n linearized polynomials:

$$\mathcal{O}\left((n-1)^2 m(\log_2(q)) + n^2(m\log_2(q)\log_2(m\log_2(q))) + n^2\right) \tag{24}$$

Since this research aims for an efficient alternative for MM in $\mathbf{F}_q^{n\times n}$ we need to be able to represent matrices in the alternative field. We say m to be equal to n in all comparisons. This way **all** elements can be represented in both $\mathbf{F}_q^{n\times n}$ and $\mathbf{F}_{q^m}$ and thus be compared. Note that not all elements of $\mathbf{F}_q^{n\times n}$ require the degree of the linearized polynomial that represents it in $\mathbf{F}_{q^m}$ to be m. Also note that by letting $m = n$, quadratic asymptotic complexities with respect to n become cubic.

4.1 Exploring Karatsuba vs Schoolbook

Both complexities, (23) and (24), share the same asymptotic theoretical complexity of $\mathcal{O}(n^2)$. In order to determine which multiplication algorithm is most suitable for competing with MM in $\mathbf{F}_q^{n\times n}$, let us explore the quantified complexities of both KA and SB in $\mathbf{F}_{q^m}$ for various values of q, and n. Appendix B contains an elaborate table that covers more values of q and n.

n	KA	SB	Diff	%Diff
2	50	14	-36	-72.0
4	344	180	-164	-47.7
8	2600	1992	-608	-23.4
16	20960	20240	-720	-3.4
32	175328	195616	20288	11.6
64	1493504	1830976	337472	22.6

(a) $q = 2$

n	KA	SB	Diff	%Diff
2	112	40	-72	-64.3
4	768	472	-296	-38.5
8	5776	4944	-832	-14.4
16	46272	48416	2144	4.6
32	384448	455744	71296	18.5
64	3253248	4182144	928896	28.6

(b) $q = 4$

n	KA	SB	Diff	%Diff
2	260	108	-152	-58.5
4	1736	1184	-552	-31.8
8	12848	11872	-976	-7.6
16	101792	112960	11168	11.0
32	838592	1041536	202944	24.2
64	7047296	9408768	2361472	33.5

(c) $q = 16$

n	KA	SB	Diff	%Diff
2	984	468	-516	-52.4
4	6268	4736	-1532	-24.4
8	45098	45221	123	0.3
16	350813	416229	65416	18.6
32	2852712	3745412	892700	31.3
64	23733952	33200648	9466696	39.9

(d) $q = 4093$

n	KA	SB	Diff	%Diff
2	1183	570	-613	-51.8
4	7494	5723	-1771	-23.6
8	53718	54349	631	1.2
16	416868	498379	81511	19.6
32	3384148	4472105	1087957	32.1
64	28119298	39554879	11435581	40.7

(e) $q = 16389$

n	KA	SB	Diff	%Diff
2	1387	675	-712	-51.3
4	8743	6735	-2008	-23.0
8	62478	63678	1200	1.9
16	483860	582130	98270	20.3
32	3922406	5211524	1289118	32.9
64	32556446	46010317	13453871	41.3

(f) $q = 65521$

Table 2: quantified complexities for KA and Schoolbook for various n and q common in the field of cryptography

4.2 Concluding Karatsuba vs Schoolbook

We see that SB is faster for small n, regardless of q. We also see that with increasing n, KA becomes increasingly superior over SB. This increase is relatively bigger for larger q. We can conclude KA to be the preferred algorithm for multiplication in $\mathbf{F}_{q^m}$ for the comparison with MM.

5 Karatsuba vs Matrix multiplication

As mentioned, a linearized polynomial and a matrix can be used to represent the same element:

$$A^{m\times m} \in \mathbf{F}_q^{m\times m} \quad \equiv \quad f(x) \in \mathbf{F}_{q^m} \text{ where } deg_q(f(x)) < m$$

Before comparing computational complexities, let us define the scope of the comparison. Although most cryptographic algorithms, as of today, make use of MM, we are not considering KA for linearized polynomials as a multiplication algorithm to plug in place of MM by means of transformations back and forth. We do therefore not explore transformations between the two discussed 'worlds'. The scope of this research is the comparison of:

$$f_1 \cdot f_2, \text{ where } f \in \mathbf{F}_{q^m} \text{ by means of KA} \tag{25}$$

and

$$A_1 \cdot A_2, \text{ where } A \in \mathbf{F}_q^{m\times m} \text{ by means of naive matrix multiplication} \tag{26}$$

5.1 Complexity of naive Matrix multiplication

In order to determine the complexity of naive MM, we define the following complexities:

$$\begin{array}{rll} \text{addition} & \text{in } \mathbf{F}_q & \mathcal{O}(\log_2(q)) \\ \text{multiplication} & \text{in } \mathbf{F}_q & \mathcal{O}(\log_2(q)\log_2(\log_2(q))) \end{array}$$

And the following number of operations:

$$\begin{array}{rl} n^3 & \text{multiplications} \\ n^3 - n^2 & \text{additions} \end{array}$$

Combining these definitions we can state the complexity of naive MM to be:

$$\mathcal{O}(n^3\log_2(q) + n^3\log_2(q)\log_2(\log_2(q)) - n^2\log_2(q)) \tag{27}$$

With the earlier found complexity of KA for linearized polynomials (23), we now can compare.

5.2 Exploring Karatsuba vs Matrix multiplication

As can be seen in table 3, with increasing n, the number of operations significantly start to differ. In order for sensible conclusions to be drawn, the operation complexity must to be taken into account.

	KA in $\mathbf{F}_{3^n}$			Naive in $\mathbf{F}_3^{n\times n}$	
n	#MUL	#ADD	#EXP	#MUL	#ADD
2	6	12	12	8	4
3	10	26	24	27	18
5	21	69	60	125	100
7	36	132	112	343	294
11	78	318	264	1331	1210

Table 3: comparison for number of operations KA and MM for small primes

Table 4 shows the quantified complexity of both algorithms for 6 different primes q commonly used in the field of cryptography. Three of which are on the low side and the other three are on the high side. Note that for all values of q, n and m, MM is superior over KA. Note that this superiority decreases with increase of n. This decrease is faster for higher n. However, even on the high end of the spectrum ($q = 65521$, $n = 64$), KA is 35,8% slower than naive MM. Appendix B contains an elaborate table that covers more values of q and n.

n	KA	MM	Diff	%Diff
2	50	4	-46	-92.0
4	344	48	-296	-86.0
8	2600	448	-2152	-82.8
16	20960	3840	-17120	-81.7
32	175328	31744	-143584	-81.9
64	1493504	258048	-1235456	-82.7

(a) $q = 2$

n	KA	MM	Diff	%Diff
2	112	24	-88	-78.6
4	768	224	-544	-70.8
8	5776	1920	-3856	-66.8
16	46272	15872	-30400	-65.7
32	384448	129024	-255424	-66.4
64	3253248	1040384	-2212864	-68.0

(b) $q = 4$

n	KA	MM	Diff	%Diff
2	260	80	-180	-69.2
4	1736	704	-1032	-59.4
8	12848	5888	-6960	-54.2
16	101792	48128	-53664	-52.7
32	838592	389120	-449472	-53.6
64	7047296	3129344	-3917952	-55.6

(c) $q = 16$

n	KA	MM	Diff	%Diff
2	984	392	-592	-60.2
4	6268	3328	-2940	-46.9
8	45098	27398	-17700	-39.2
16	350813	222262	-128551	-36.6
32	2852712	1790384	-1062328	-37.2
64	23733952	14372227	-9361725	-39.4

(d) $q = 4093$

n	KA	MM	Diff	%Diff
2	1183	482	-701	-59.3
4	7494	4083	-3411	-45.5
8	53718	33564	-20154	-37.5
16	416868	272100	-144768	-34.7
32	3384148	2191137	-1193011	-35.3
64	28119298	17586444	-10532854	-37.5

(e) $q = 16389$

n	KA	MM	Diff	%Diff
2	1387	575	-812	-58.5
4	8743	4863	-3880	-44.4
8	62478	39934	-22544	-36.1
16	483860	323575	-160285	-33.1
32	3922406	2604986	-1317420	-33.6
64	32556446	20905427	-11651019	-35.8

(f) $q = 65521$

Table 4: quantified complexities for KA and MM for various n and q common in the field of cryptography

6 Conclusion

In this research we implemented the Karatsuba Algorithm for same degree linearized polynomials. We analyzed the complexity and compared that to Schoolbook multiplication for same degree linearized polynomials. This comparison showed the Karatsuba algorithm to be faster for linearized polynomials of degree $n > 19$ independent of q and for even lower degrees n when q gets larger. We also compared to naive matrix multiplication. Matrix multiplication is superior over the Karatsuba Algorithm in all cases.

7 Discussion

This research implemented single iteration Karatsuba as contender for matrix multiplication. Recursive Karatsuba (divide & conquer) fell out of the scope of this research. Although this paper did not conclude in an enticing alternative for matrix multiplication, perhaps the exploration of recursive Karatsuba could.

As mentioned, this research assumes m to be equal to n. Using cases in which the 'dimension' of the field of linearized polynomials m is strictly smaller than matrix size n has potency of being in favor of Karatsuba.

This research assumed and compared to naive matrix multiplication. There are known to be more efficient matrix multiplication algorithms, of which the Strassen algorithm is best known. Future research could incorporate other algorithms.

Lastly, this research did not consider real-world implementations of the algorithms, solely theory. Actual running time might show different results, especially when utilizing hardware in favorable ways.

References

[1] Shuhong Gao. *Normal bases over finite fields.* University of Waterloo Waterloo, Canada, 1993.

[2] A. Karatsuba and Y. Ofman. *Multiplication of Multidigit Numbers on Automata.* 7:595–596, 1963.

[3] Neal Koblitz. *Elliptic Curve Cryptosystems.* 48(177):203–209, Jan 1987.

[4] Rudolf Lidl and Harald Niederreiter. *Finite fields.* Number 20. Cambridge university press, 1997.

[5] Sudhanshu Mishra and Manoranjan Pradhan. Implementation of karatsuba algorithm using polynomial multiplication. *Indian Journal of Computer Science and Engineering*, 3, Feb 2012.

[6] National Institute of Standards and Technology. *Post-Quantum Cryptography*, Jun 2023.

[7] Oystein Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35(3):559–584, 1933.

[8] R. L. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM*, 21(2):120–126, feb 1978.

[9] André Weimerskirch and Christof Paar. *Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive*, 2006.

A Karatsuba algorithm for q-degree 3 linearized polynomial

Then, following the earlier given definition of multiplication for linearized polynomials (2), the product $C(x) = A(x)B(x)$ can be determined in the following manner:

$$
\begin{aligned}
C(x) = a_3 b_3^{[3]} x^{[6]} &+ a_3 b_2^{[3]} x^{[5]} + a_3 b_1^{[3]} x^{[4]} + a_3 b_0^{[3]} x^{[3]} \\
&+ a_2 b_3^{[2]} x^{[5]} + a_2 b_2^{[2]} x^{[4]} + a_2 b_1^{[2]} x^{[3]} + a_2 b_0^{[2]} x^{[2]} \\
&+ a_1 b_3^{[1]} x^{[4]} + a_1 b_2^{[1]} x^{[3]} + a_1 b_1^{[1]} x^{[2]} + a_1 b_0^{[1]} x^{[1]} \\
&+ a_0 b_3^{[0]} x^{[3]} + a_0 b_2^{[0]} x^{[2]} + a_0 b_1^{[0]} x^{[1]} + a_0 b_0^{[0]} x^{[0]} \\
\\
&= a_3 b_3^{[3]} x^{[6]} + (a_3 b_2^{[3]} + a_2 b_3^{[2]}) x^{[5]} + (a_3 b_1^{[3]} + a_2 b_2^{[2]} + a_1 b_3^{[1]}) x^{[4]} \\
&\quad + (a_3 b_0^{[3]} + a_2 b_1^{[2]} + a_1 b_2^{[1]} + a_0 b_3^{[0]}) x^{[3]} + (a_2 b_0^{[2]} + a_1 b_1^{[1]} + a_0 b_2^{[0]}) x^{[2]} \\
&\quad + (a_1 b_0^{[1]} + a_0 b_1^{[0]}) x^{[1]} + a_0 b_0^{[0]} x^{[0]}
\end{aligned}
$$

The coefficients of $x^{[1]}$, (part of) $x^{[2]}$, (part of) $x^{[3]}$, (part of) $x^{[4]}$, and $x^{[5]}$ in the resulting polynomial $C(x)$ can be written as:

$$
\begin{aligned}
(a_1 b_0^{[1]} + a_0 b_1^{[0]}) &= ((a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) - a_0 b_0^{[1]} - a_1 b_1^{[0]}) \\
(a_2 b_0^{[2]} + a_0 b_2^{[0]}) &= ((a_0 + a_2)(b_0^{[2]} + b_2^{[0]}) - a_0 b_0^{[2]} - a_2 b_2^{[0]}) \\
(a_3 b_0^{[3]} + a_0 b_3^{[0]}) &= ((a_0 + a_3)(b_0^{[3]} + b_3^{[0]}) - a_0 b_0^{[3]} - a_3 b_3^{[0]}) \\
(a_2 b_1^{[2]} + a_1 b_2^{[1]}) &= ((a_1 + a_2)(b_1^{[2]} + b_2^{[1]}) - a_1 b_1^{[2]} - a_2 b_2^{[1]}) \\
(a_3 b_1^{[3]} + a_1 b_3^{[1]}) &= ((a_1 + a_3)(b_1^{[3]} + b_3^{[1]}) - a_1 b_1^{[3]} - a_3 b_3^{[1]}) \\
(a_3 b_2^{[3]} + a_2 b_3^{[2]}) &= ((a_2 + a_3)(b_2^{[3]} + b_3^{[2]}) - a_2 b_2^{[3]} - a_3 b_3^{[2]})
\end{aligned}
$$

Let there be the following auxiliary variables:

$$
\begin{aligned}
&D_{0^0} = a_0 b_0^{[0]}, D_{0^1} = a_0 b_0^{[1]}, D_{0^2} = a_0 b_0^{[2]}, D_{0^3} = a_0 b_0^{[3]} \\
&D_{1^0} = a_1 b_1^{[0]}, D_{1^1} = a_1 b_1^{[1]}, D_{1^2} = a_1 b_1^{[2]}, D_{1^3} = a_1 b_1^{[3]} \\
&D_{2^0} = a_2 b_2^{[0]}, D_{2^1} = a_2 b_2^{[1]}, D_{2^2} = a_2 b_2^{[2]} D_{2^3} = a_2 b_2^{[3]} \\
&D_{3^0} = a_3 b_3^{[0]}, D_{3^1} = a_3 b_3^{[1]}, D_{3^2} = a_3 b_3^{[2]} D_{3^3} = a_3 b_3^{[3]} \\
&D_{0,1} = (a_0 + a_1)(b_0^{[1]} + b_1^{[0]}) \\
&D_{0,2} = (a_0 + a_2)(b_0^{[2]} + b_2^{[0]}) \\
&D_{0,3} = (a_0 + a_3)(b_0^{[3]} + b_3^{[0]}) \\
&D_{1,2} = (a_1 + a_2)(b_1^{[2]} + b_2^{[1]}) \\
&D_{1,3} = (a_1 + a_3)(b_1^{[3]} + b_3^{[1]}) \\
&D_{2,3} = (a_2 + a_3)(b_2^{[3]} + b_3^{[2]})
\end{aligned}
$$

Then:

$$
\begin{aligned}
C(x) = D_{3^3} x^{[6]} + & \\
& (D_{2,3} - D_{2^3} - D_{3^2}) x^{[5]} + \\
& (D_{1,3} - D_{1^3} - D_{3^1} + D_{2^2}) x^{[4]} + \\
& ((D_{0,3} - D_{0^3} - D_{3^0}) + (D_{1,2} - D_{1^2} - D_{2^1})) x^{[3]} + \\
& (D_{0,2} - D_{2^0} - D_{0^2} + D_{1^1}) x^{[2]} + \\
& (D_{0,1} - D_{0^1} - D_{1^0}) x^{[1]} + \\
& D_{0^0} x^{[0]}
\end{aligned}
$$

B Complexities

The following table contains the complexity for different values of q and n. The **KA** and **SB** columns display the complexity of Karatsuba and Schoolbook respectively. The **SB-KA** and the **%** column that follows that, show the difference between SB and KA with respect to KA, absolute and percentage-wise. The **MM** column displays the complexity of naive matrix multiplication. The **MM-KA** and the **%** column that follows that show the difference between MM and KA with respect to KA, again, absolute and percentage-wise.

q	n	KA	SB	SB-KA	%	MM	MM-KA	%
2	2	50	14	-36	-72.0	4	-46	-92.0
	4	344	180	-164	-47.7	48	-296	-86.0
	8	2600	1992	-608	-23.4	448	-2152	-82.8
	16	20960	20240	-720	-3.4	3840	-17120	-81.7
	32	175328	195616	20288	11.6	31744	-143584	-81.9
	64	1493504	1830976	337472	22.6	258048	-1235456	-82.7
	128	12813440	16760960	3947520	30.8	2080768	-10732672	-83.8
	256	110070272	150929664	40859392	37.1	16711680	-93358592	-84.8
	512	943986176	1341915648	397929472	42.2	133955584	-810030592	-85.8
	1048	8672291052	12698621885	4026330833	46.4	1149924288	-7522366764	-86.7
4	2	112	40	-72	-64.3	24	-88	-78.6
	4	768	472	-296	-38.5	224	-544	-70.8
	8	5776	4944	-832	-14.4	1920	-3856	-66.8
	16	46272	48416	2144	4.6	15872	-30400	-65.7
	32	384448	455744	71296	18.5	129024	-255424	-66.4
	64	3253248	4182144	928896	28.6	1040384	-2212864	-68.0
	128	27740416	37699840	9959424	35.9	8355840	-19384576	-69.9
	256	236983296	335348224	98364928	41.5	66977792	-170005504	-71.7
	512	2022452224	2952004608	929552384	46.0	536346624	-1486105600	-73.5
	1048	18496703001	27698190651	9201487650	49.7	4601893760	-13894809241	-75.1
16	2	260	108	-152	-58.5	80	-180	-69.2
	4	1736	1184	-552	-31.8	704	-1032	-59.4
	8	12848	11872	-976	-7.6	5888	-6960	-54.2
	16	101792	112960	11168	11.0	48128	-53664	-52.7
	32	838592	1041536	202944	24.2	389120	-449472	-53.6
	64	7047296	9408768	2361472	33.5	3129344	-3917952	-55.6
	128	59740928	83771904	24030976	40.2	25100288	-34640640	-58.0
	256	507783680	737739776	229956096	45.3	201064448	-306719232	-60.4
	512	4314389504	6440617984	2126228480	49.3	1609564160	-2704825344	-62.7
	1048	39299846499	59999373367	20699526868	52.7	13807877888	-25491968611	-64.9
31	2	337	145	-192	-57.0	111	-226	-67.1
	4	2232	1560	-672	-30.1	969	-1263	-56.6
	8	16429	15471	-958	-5.8	8075	-8354	-50.8
	16	129687	146108	16421	12.7	65872	-63815	-49.2
	32	1065584	1339855	274271	25.7	532050	-533534	-50.1
	64	8936354	12053092	3116738	34.9	4276695	-4659659	-52.1
	128	75625365	106958498	31333133	41.4	34294731	-41330634	-54.7
	256	641861499	939365625	297504126	46.4	274682526	-367178973	-57.2
	512	5446677244	8182193649	2735516405	50.2	2198758926	-3247918318	-59.6
	1048	49556809471	76071955705	26515146234	53.5	18861783586	-30695025885	-61.9
127	2	512	230	-282	-55.1	184	-328	-64.1
	4	3340	2416	-924	-27.7	1590	-1750	-52.4
	8	24365	23575	-790	-3.2	13167	-11198	-46.0
	16	191214	220213	28999	15.2	107132	-84082	-44.0
	32	1564584	2003330	438746	28.0	864213	-700371	-44.8

	64	13078958	17910499	4831541	36.9	6942331	-6136627	-46.9
	128	110391266	158150270	47759004	43.3	55653155	-54738111	-49.6
	256	934835817	1383297950	448462133	48.0	445683255	-489152562	-52.3
	512	7917356589	12007780192	4090423603	51.7	3567298088	-4350058501	-54.9
	1048	71909038804	111304051268	39395012464	54.8	30600422610	-41308616194	-57.4
251	2	601	274	-327	-54.4	222	-379	-63.1
	4	3895	2851	-1044	-26.8	1910	-1985	-51.0
	8	28316	27656	-660	-2.3	15794	-12522	-44.2
	16	221733	257345	35612	16.1	128397	-93336	-42.1
	32	1811490	2334514	523024	28.9	1035340	-776150	-42.8
	64	15124903	20825483	5700580	37.7	8315374	-6809529	-45.0
	128	127535777	183563173	56027396	43.9	66653605	-60882172	-47.7
	256	1079132300	1603218882	524086582	48.6	533751264	-545381036	-50.5
	512	9132897922	13899578519	4766680597	52.2	4272099806	-4860798116	-53.2
	1048	82895128418	128699070826	45803942408	55.3	36645781808	-46249346610	-55.8
256	2	604	276	-328	-54.3	224	-380	-62.9
	4	3912	2864	-1048	-26.8	1920	-1992	-50.9
	8	28432	27776	-656	-2.3	15872	-12560	-44.2
	16	222624	258432	35808	16.1	129024	-93600	-42.0
	32	1818688	2344192	525504	28.9	1040384	-778304	-42.8
	64	15184512	20910592	5726080	37.7	8355840	-6828672	-45.0
	128	128035072	184304640	56269568	43.9	66977792	-61057280	-47.7
	256	1083333120	1609631744	526298624	48.6	536346624	-546986496	-50.5
	512	9168274432	13954715648	4786441216	52.2	4292870144	-4875404288	-53.2
	1048	83214772695	129205829166	45991056471	55.3	36823936512	-46390836183	-55.7
2039	2	886	418	-468	-52.8	348	-538	-60.7
	4	5663	4252	-1411	-24.9	2961	-2702	-47.7
	8	40832	40727	-105	-0.3	24392	-16440	-40.3
	16	318069	375693	57624	18.1	197956	-120113	-37.8
	32	2588969	3386224	797255	30.8	1594906	-994063	-38.4
	64	21555650	30055565	8499915	39.4	12804285	-8751365	-40.6
	128	181345292	263839503	82494211	45.5	102614404	-78730888	-43.4
	256	1531462824	2296525779	765062955	50.0	821635712	-709827112	-46.3
	512	12939224354	19853233240	6914008886	53.4	6575967616	-6363256738	-49.2
	1048	117263243027	183358053040	66094810013	56.4	56406735256	-60856507771	-51.9
4093	2	984	468	-516	-52.4	392	-592	-60.2
	4	6268	4736	-1532	-24.4	3328	-2940	-46.9
	8	45098	45221	123	0.3	27398	-17700	-39.2
	16	350813	416229	65416	18.6	222262	-128551	-36.6
	32	2852712	3745412	892700	31.3	1790384	-1062328	-37.2
	64	23733952	33200648	9466696	39.9	14372227	-9361725	-39.4
	128	199550825	291140986	91590161	45.9	115174406	-84376419	-42.3
	256	1684348673	2531934722	847586049	50.3	922181616	-762167057	-45.2
	512	14224630791	21871958972	7647328181	53.8	7380598384	-6844032407	-48.1
	1048	128860229796	201868316337	73008086541	56.7	63308232880	-65551996916	-50.9
16389	2	1183	570	-613	-51.8	482	-701	-59.3
	4	7494	5723	-1771	-23.6	4083	-3411	-45.5
	8	53718	54349	631	1.2	33564	-20154	-37.5
	16	416868	498379	81511	19.6	272100	-144768	-34.7
	32	3384148	4472105	1087957	32.1	2191137	-1193011	-35.3
	64	28119298	39554879	11435581	40.7	17586444	-10532854	-37.5
	128	236175965	346237093	110061128	46.6	140920942	-95255023	-40.3
	256	1991732478	3006543936	1014811458	51.0	1128285073	-863447405	-43.4
	512	16807637297	25938515383	9130878086	54.3	9029950721	-7777686576	-46.3
	1048	152152972999	239127599600	86974626601	57.2	77455027831	-74697945168	-49.1

65521	2	1387	675	-712	-51.3	575	-812	-58.5
	4	8743	6735	-2008	-23.0	4863	-3880	-44.4
	8	62478	63678	1200	1.9	39934	-22544	-36.1
	16	483860	582130	98270	20.3	323575	-160285	-33.1
	32	3922406	5211524	1289118	32.9	2604986	-1317420	-33.6
	64	32556446	46010317	13453871	41.3	20905427	-11651019	-35.8
	128	273203450	402138028	128934578	47.2	167505559	-105697891	-38.7
	256	2302277782	3487553430	1185275648	51.5	1341093029	-961184753	-41.7
	512	19415632121	30055968481	10640336360	54.8	10732938452	-8682693669	-44.7
	1048	175658002840	276820659349	101162656509	57.6	92061785874	-83596216966	-47.6