*Int. J. Communications, Network and System Sciences*, 2012, 5, 437-445
http://dx.doi.org/10.4236/ijcns.2012.58054 Published Online August 2012 (http://www.SciRP.org/journal/ijcns)

# Large-Integer Multiplication Based on Homogeneous Polynomials

**Boris S. Verkhovsky**
Computer Science Department, New Jersey Institute of Technology, Newark, USA
Email: verb@njit.edu

Received June 17, 2012; revised July 28, 2012; accepted August 6, 2012

## ABSTRACT

Several algorithms based on homogeneous polynomials for multiplication of large integers are described in the paper. The homogeneity of polynomials provides several simplifications: reduction of system of equations and elimination of necessity to evaluate polynomials in points with larger coordinates. It is demonstrated that a two-stage implementation of the proposed and Toom-Cook algorithms asymptotically require twice as many standard multiplications than their direct implementation. A multistage implementation of these algorithms is also less efficient than their direct implementation. Although the proposed algorithms as well as the corresponding Toom-Cook algorithms require numerous algebraic additions, the Generalized Horner rule for evaluation of homogeneous polynomials, provided in the paper, decrease this number twice.

**Keywords:** Homogeneous Polynomials; Toom-Cook Algorithm; Multidigit Integers; Multi-Stage Multiplication; Generalized Horner Rule; Large-Integer Multiplication

## 1. Introduction and Basic Definitions

Crypto-immunity of various protocols of secure communication over open channels is based on modular arithmetic of large integers with hundreds of decimal digits. Multiplications and exponentiations of large integers are essential operations in this arithmetic. Yet, standard programming libraries in general-purpose computers handle multiplication of integers $A$ and $B$ if the number of decimal digits in each does not exceed $m$. Such integers we will refer to as *standard* integers. For instance, if a computer cannot multiply integers larger than $10^{30}$ without a specially-written program, then in this case $m = 30$.

The first papers on multiplication of large integers were published by Karatsuba-Ofman [1] and by Toom [2]. Several years later Toom's scheme was improved by Cook {see [3,4]}. Analysis of computational complexity of Toom-Cook algorithm (TCA) is provided in [5] and theoretical foundation for efficient multiplication of large integers is discussed in [6]. An efficient implementation of the TCA in cryptographic systems is described in several patents [7]. Analysis of computational complexity of the TCA and its lower bound is provided in [8].

A special case of the TCA, where one multiplier is significantly larger than another, is considered in [9].

Consider two $nm$-digit-large integers $A = \overline{a_{n-1}\cdots a_2 a_1 a_0}$ and $B = \overline{b_{n-1}\cdots b_2 b_1 b_0}$, where every part $a_k$ and $b_k$ is a $m$-decimal-digit large *standard integer* {SI, for short}. Let us represent $A$ and $B$ as

$$A = \sum_{k=0}^{n-1} \left(10^m\right)^k a_k; \quad (1.1)$$

and

$$B = \sum_{k=0}^{n-1} \left(10^m\right)^k b_k. \quad (1.2)$$

Therefore, the product $C = AB$ is expressed as

$$C = \sum_{s=0}^{2(n-1)} \left(10^m\right)^s c_s; \quad (1.3)$$

where $c_{2n-2}, c_{2n-3}, \cdots, c_1, c_0$ are $2n-1$ unknown coefficients.

In order to compute the product $C$, these coefficients must be determined.

***Example* 1.1:** Suppose we need to multiply two integers

$$A = 385{,}495{,}374{,}109;$$

and

$$B = 608{,}348{,}696{,}284;$$

using a computing device that cannot multiply integers of order higher than $O\left(10^3\right)$. Therefore, in this example $m = 3$ and we split $A$ and $B$ into $n = 4$ parts, where

$$a_3 = 385;\ a_2 = 495;\ a_1 = 374;\ a_0 = 109;$$
$$b_3 = 608;\ b_2 = 348;\ b_1 = 696;\ b_0 = 284.$$

The algorithm provided in Section 2 demonstrates how to solve this problem by using a minimal number of multiplications of *m*-digit SIs.

## 2. Multiplication *C* = *AB* Based on Homogeneous Polynomials

Consider two *n*-th degree homogeneous polynomials of two integer variables *x* and *y*:

$$A_n(x,y) := \sum_{i=0}^{n} a_i x^i y^{n-i}; \quad (2.1)$$

$$B_n(x,y) := \sum_{i=0}^{n} b_i x^i y^{n-i}. \quad (2.2)$$

Let

$$C_{2n}(x,y) := A_n(x,y) B_n(x,y) = \sum_{k=0}^{2n} c_k x^k y^{2n-k}. \quad (2.3)$$

***Remark* 2.1:** All coefficients in polyno-mials $A_n(x,y)$ and $B_n(x,y)$ are *inputs*, and the coefficients in $C_{2n}(x,y)$ are *outputs*.

For short, the multiplication algorithm, based on homogeneous polynomials (HP), provided below is called the AHP.

First of all, definition (2.3) implies that

$$c_{2n} = a_n b_n; \text{ and } c_0 = a_0 b_0. \quad (2.4)$$

Computation of the remaining $2n-1$ coefficients in (2.3) is described in the algorithm. Prior to that, let us modify Equation (2.3) for integers $|x| \ge 1; |y| \ge 1$. Consider

$$M_{2(n-1)}(x,y) := \left[C_{2n}(x,y) - c_{2n}x^{2n} - c_0 y^{2n}\right] / xy = \sum_{k=1}^{2n-1} c_k x^{k-1} y^{2n-1-k}. \quad (2.5)$$

As is shown in Section 3, we can easily separate "*odd*" $c_1, c_3, \cdots, c_{2n-1}$ and "*even*" $c_2, c_4, \cdots, c_{2n-2}$ unknown coefficients.

The following properties of homogeneous polynomials imply certain limitations on choice of evaluation points:

*Property* 1: If *n* is a degree of homogeneity of $H(x,y)$ and *g* is a non-zero real number, then

$$H(gx, gy) = g^n H(x,y). \quad (2.6)$$

Therefore, if $g = -1$, then

$$H(-x,-y) = (-1)^n H(x,y). \quad (2.7)$$

*Property* 2: If the degree of homogeneity is *even*, then $H(-x,y) = H(x,-y)$;

otherwise $H(-x,y) = -H(x,-y)$. (2.8)

***Corollary* 1:** Definition (2.3) implies that

$$C(-x,y) = C(x,-y); \quad (2.9)$$

since for every integer *n* $C_{2n}(x,y)$ has an even degree of homogeneity.

***Corollary* 2:** Identity (2.9) implies that it is not advantageous to consider, for instance, both $C(p,-q)$ and $C(-p, q)$; neither it is advantageous to consider $C(p, q)$ and $C(gp, gq)$ for any non-zero integer *g* (2.6). Therefore, in this paper are considered only relatively prime pairs of integers *p* and *q*.

## 3. Separation of "Even" and "Odd" Coefficients in AHP

**Step 3.1:** Compute *sums*

$$S_{2(n-1)}(p,q) := \left[M_{2(n-1)}(p,q) + M_{2(n-1)}(p,-q)\right] / 2pq; \quad (3.1)$$

for the first $n-1$ relatively prime pairs of integers

$$(p, q) = \{(2,1);(1,2);(3,1);(1,3);(3,2); (2,3);(4,1);(1,4);\cdots\}. \quad (3.2)$$

***Remark* 3.1:** Using (3.1) and (3.2), we create and solve $n-1$ equations with $n-1$ "even" unknowns $c_2, c_4, \cdots, c_{2(n-1)}$.

**Step 3.2:** Compute *differences*

$$D_{2(n-1)}(p,q) := \left[M_{2(n-1)}(p,q) - M_{2(n-1)}(p,-q)\right] / 2(pq)^2 \quad (3.3)$$

for the same pairs

$$(p, q) = \{(2,1);(1,2);(3,1);(1,3);(3,2); (2,3);(4,1);(1,4);\cdots\}.$$

***Remark* 3.2:** As a result, in (3.3), we create $n-1$ equations with *n* unknowns $c_1, c_3, \cdots, c_{2n-1}$.

Since by this time the values of all "even" coefficients $c_2, c_4, \cdots, c_{2n-2}$ are already computed, we use $M_{2(n-1)}(1,1)$ (3.5) as the *n*-th equation for computation of *n* "odd" coefficients.

The following example illustrates a slightly different approach to separate "even" and "odd" variables.

### 3.1. Separation of Unknowns: *n* = 5

First, compute $S_8(p,q)$ {see (3.1)} for $(p,q) = \{(1,1);(2,1);(1,2)\}$ and from three equations find $c_2, c_4$ and $c_6$; second, compute $D_8(p,q)$ {see (3.3)} for $(p,q) = \{(1,1);(2,1);(1,2)\}$ and derive three equations with four "odd" unknowns $c_1, c_3, c_5, c_7$. As the fourth ("missing") equation, we consider

$$M_8(3,1) := 3^7 c_7 + 3^6 c_6 + \cdots + 3c_2 + c_1; \quad (3.4)$$

{see (2.5)}. After all "even" coefficients are computed, let

$$Z := M_8(3,1) - 3^2\left(3^4 c_6 + 3^2 c_4 + c_2\right). \quad (3.5)$$

Finally we derive the *fourth* equation

$$3^7 c_7 + 3^5 c_5 + 3^3 c_3 + 3c_1 = Z. \quad (3.6)$$

## 3.2. AHP for Multiplication of Triple-Large Integers

Let us consider a multiplication of triple-large integers; let

$$A(x,y) = a_2 x^2 + a_1 xy + a_0 y^2;$$

and

$$B(x,y) = b_2 x^2 + b_1 xy + b_0 y^2;$$

therefore

$$\begin{aligned} C(x,y) &= A(x,y)B(x,y) \\ &= c_4 x^4 + c_3 x^3 y + c_2 x^2 y^2 + \cdots + c_0 y^4. \end{aligned} \quad (3.7)$$

**Step 3.1:**

$$c_4 := a_2 b_2 \left\{= C(1,0) = A(1,0)B(1,0)\right\}; \quad (3.8)$$

**Step 3.2:**

$$c_0 := a_0 b_0 \left\{= C(0,1) = A(0,1)B(0,1)\right\}; \quad (3.9)$$

**Step 3.3:**

$$\begin{aligned} F &:= (4a_2 + 2a_1 + a_0)(4b_2 + 2b_1 + b_0) \\ &\left\{= C(2,1) = A(2,1)B(2,1)\right. \\ &\left.= 16c_4 + 8c_3 + 4c_2 + 2c_1 + c_0\right\}; \end{aligned} \quad (3.10)$$

**Step 3.4:**

$$\begin{aligned} G &:= (a_2 + a_1 + a_0)(b_2 + b_1 + b_0) \\ &\left\{= C(1,1) = A(1,1)B(1,1) = c_4 + c_3 + \cdots + c_0\right\}; \end{aligned} \quad (3.11)$$

**Step 3.5:**

$$\begin{aligned} H &:= (a_2 + 2a_1 + 4a_0)(b_2 + 2b_1 + 4b_0) \\ &\left\{= C(1,2) = A(1,2)B(1,2) = c_4 + 2c_3 + \cdots + 16c_0\right\}; \end{aligned} \quad (3.12)$$

**Step 3.6:**

$$K := G - c_4 - c_0 \left\{= M(1,1) = c_3 + c_2 + c_1\right\}; \quad (3.13)$$

**Step 3.7:**

$$L := \left[F - 16c_4 - c_0\right]/2 \left\{= M(2,1) = 4c_3 + 2c_2 + c_1\right\}; \quad (3.14)$$

**Step 3.8:**

$$M := \left[H - c_4 - 16c_0\right]/2 \left\{= M(1,2) = c_3 + 2c_2 + 4c_1\right\}; \quad (3.15)$$

***Remark* 3.3:** From the system of linear Equations (3.13)-(3.15) we determine $c_3, c_2$ and $c_1$.

**Step 3.9:**

$$\begin{aligned} &N := (L - M)/3 = c_3 - c_1; \\ &\text{and } P := L - 2K = 2c_3 - c_1. \end{aligned} \quad (3.16)$$

**Step 3.10:**

$$c_3 = P - N;\ c_1 = c_3 - N;\ c_2 = K - c_1 - c_3. \quad (3.17)$$

The algorithm described in (3.8)-(3.17) requires 24 algebraic additions.

## 4. Reduction of Algebraic Additions

Let us consider a multiplication of two quatro-large integers

$$A = \overline{a_3 a_2 a_1 a_0}; \text{ and } B = \overline{b_3 b_2 b_1 b_0};$$

where every part $a_k$ and $b_k$ is a *m*-decimal-digit large SI [2].

Let us represent *A* and *B* as

$$A = \left(10^m\right)^3 a_3 + \left(10^m\right)^2 a_2 + 10^m a_1 + a_0; \quad (4.1)$$

and

$$B = 10^{3m} b_3 + 10^{2m} b_2 + 10^m b_1 + b_0. \quad (4.2)$$

Therefore, the product *C* = *AB* can be expressed as

$$C = 10^{6m} c_6 + 10^{5m} c_5 + \cdots + 10^m c_1 + c_0; \quad (4.3)$$

where *seven* coefficients $c_6, c_5, \cdots, c_1, c_0$ must be determined.

The drawback of the TCA and AHP algorithms is the large number of required algebraic additions. The following algorithm shows how to decrease *twice* the number of these additions.

**Step 4.0:**

$$c_0 := a_0 b_0;\ c_6 := a_3 b_3; \quad (4.4)$$

**Step 4.1:**

$$\begin{aligned} &A_1 := a_3 + a_1;\ B_1 := b_3 + b_1; \\ &A_0 := a_2 + a_0;\ B_0 := b_2 + b_0; \end{aligned} \quad (4.5)$$

**Step 4.2:**

$$\begin{aligned} &C_1 := (A_1 + A_0)(B_1 + B_0); \\ &C_{-1} := (-A_1 + A_0)(-B_1 + B_0); \end{aligned} \quad (4.6)$$

**Step 4.3:**

$$\begin{aligned} &A_3 := 4a_3 + a_1;\ A_2 := 4a_2 + a_0; \\ &B_3 := 4b_3 + b_1;\ B_2 := 4b_2 + b_0; \end{aligned} \quad (4.7)$$

**Step 4.4:**

$$\begin{aligned} &C_2 := (2A_3 + A_2)(2B_3 + B_2); \\ &C_{-2} := (-2A_3 + A_2)(-2B_3 + B_2); \end{aligned} \quad (4.8)$$

**Step 4.5:**

$$\begin{aligned} &A_5 := 8a_3 + A_1;\ A_4 := 8a_2 + A_0; \\ &B_5 := 8b_3 + B_1;\ B_4 := 8b_2 + B_0; \end{aligned} \quad (4.9)$$

**Step 4.6:**

$$C_3 := (3A_5 + A_4)(3B_5 + B_4); \quad C_{-3} := (-3A_5 + A_4)(-3B_5 + B_4); \tag{4.10}$$

***Remark* 4.1:** For every $k$ the variables $A_k$ and $B_k$ are used twice {see (4.6)-(4.10)}. In order to decrease the amount of computation, we pre-compute them only once. Therefore, we reduce twice the number of algebraic additions in (4.6)-(4.10).

**Step 4.7:**

$$E_1 := (C_1 + C_{-1})/2 - c_0; \quad E_2 := (C_2 + C_{-2} - 2c_0)/8; \quad E_3 := (C_3 + C_{-3} - 2c_0)/18; \tag{4.11}$$

**Step 4.8:**

$$F_2 := (E_2 - E_1)/3; \quad F_3 := (E_3 - E_1)/8; \tag{4.12}$$

**Step 4.9:**

$$c_4 := F_2 - 5c_6; \quad c_2 := E_1 - c_4 - c_6; \tag{4.13}$$

**Step 4.10:**

$$e_1 := (C_1 - C_{-1})/2; \quad e_2 := (C_2 - C_{-2})/4; \quad e_3 := (C_3 - C_{-3})/6; \tag{4.14}$$

**Step 4.11:**

$$f_2 := (e_2 - e_1)/3; \quad f_3 := (e_3 - e_1)/8; \tag{4.15}$$

**Step 4.12:**

$$c_5 := (f_3 - f_2)/5; \quad c_3 := f_2 - 5c_5; \quad c_1 := e_1 - c_3 - c_5. \tag{4.16}$$

This algorithm computes the product $C = AB$ using *seven* multiplications of SIs instead of *sixteen* such multiplications as required by "grammar-school" rules. For more details on the AHP of quatro-large integers see Sections 8 and 9.

## 5. Comparison of Evaluated Polynomials in TCA vs AHP

First of all, in the TCA

$$\Psi := \{C(0), C(1), C(-1)\}; \tag{5.1}$$

are computed, and in the AHP

$$\Phi := \{C(1,0), C(0,1), C(1,1), C(1,-1)\} \tag{5.2}$$

are computed. Additional values of evaluated polynomials for $n \ge 3$ are provided in **Table 1**.

***Remark* 5.1:** Observe the fast growth of the values of evaluation points in the TCA in comparison with corresponding points in the AHP.

The sets $\Psi$ and $\Phi$ of polynomial evaluations in **Table 1** are defined in (5.1) and (5.2).

## 6. Comparison of TCA vs AHP for *n* = 6

### 6.1. AHP Framework

Compute

$$C(1,0); C(0,1); C(1,1); C(1,1); C(2,1); C(2,1); \quad C(1,2); C(1,2); C(3,1); C(3,1); C(1,3); \tag{6.1}$$

$$C(1,0) = A(1,0)B(1,0) = a_5 b_5 = c_{10}; \quad C(0,1) = A(0,1)B(0,1) = a_0 b_0 = c_0; \tag{6.2}$$

Computation of $C(1,1)$ and $C(1,-1)$ has the same complexity as $C(1)$ in the TCA; and computation of $C(2,1)$; $C(2,-1)$; $C(1,2)$ and $C(1,-2)$ has the same complexity as $C(-2)$ in the TCA {see **Table 1**). For instance,

$$\begin{aligned} C(2,1) &= A(2,1)B(2,1) \\ &= (2^5 a_5 + 2^4 a_4 + \cdots + a_0)(2^5 b_5 + 2^4 b_4 + \cdots + b_0) \\ &= 2^{10} c_{10} + 2^9 c_9 + \cdots + 2^2 c_2 + 2c_1 + c_0; \end{aligned} \tag{6.3}$$

where all coefficients are merely binary shifts. Furthermore, computation of $C(3,1)$; $C(3,-1)$; and $C(1,3)$ has the same complexity as $C(3)$ in TCA, {**Table 1**}.

### 6.2. Toom-Cook Algorithm

Compute

**Table 1. Points of polynomial evaluation in TCA and AHP.**

| Splitting in | Toom-Cook algorithms | Algorithms based on HP |
|---|---|---|
| **3 parts** | $\Psi$, $C(2), C(-2)$ | $\Phi$, $C(2,1)$ |
| **4 parts** | $\Psi$, $C(2), C(-2)$; $C(3), C(-3)$ | $\Phi$, $C(2,1)$; $C(2,-1), C(1,2)$ |
| **8 parts** | $\Psi$, $C(2), C(-2)$; $C(3), C(-3)$; $C(4), C(-4)$; $C(5), C(-5)$; $C(6), C(-6)$; $C(7), C(-7)$ | $\Phi$, $C(2,1)$; $C(2,-1), C(1,2)$; $C(1,-2), C(3,1)$; $C(3,-1), C(1,3)$; $C(1,-3), C(3,2)$; $C(3,-2), C(2,3)$ |

$$C(0);C(1);C(-1);C(2);C(-2);C(3);C(-3);\\ C(4);C(-4);C(5) \text{ and } C(-5) \quad (6.4)$$

$$C(0)=A(0)B(0)=a_0b_0=c_0; \quad (6.5)$$

$$\begin{aligned}C(1)&=A(1)B(1)\\ &=(a_5+a_4+\cdots+a_0)(b_5+b_4+\cdots+b_0)\\ &=c_{10}+c_9+\cdots+c_2+c_1+c_0;\end{aligned} \quad (6.6)$$

$$\begin{aligned}C(-1)&=A(-1)B(-1)\\ &=(-a_5+a_4-\cdots+a_0)(-b_5+b_4-\cdots+b_0)\\ &=c_{10}-c_9+c_8-\cdots-c_1+c_0;\end{aligned} \quad (6.7)$$

$$\begin{aligned}C(5)&=A(5)B(5)\\ &=(3125a_5+\cdots+5a_1+a_0)\\ &\quad\times(3125b_5+625b_4+\cdots+5b_1+b_0)\\ &=9765625c_{10}+\cdots+5c_1+c_0;\end{aligned} \quad (6.8)$$

where both $A(p)$ and $B(p)$ can be computed by Horner Rule [10].

## 7. AHP for *n* = 7

It is easy to see that

$$C(1,0)=a_7b_7=c_{14}; \text{ and } C(0,1)=a_0b_0=c_0, \quad (7.1)$$

*i.e.*, we need to compute *thirteen* remaining coefficients $c_{13},c_{12},\cdots,c_1$.

Let $$M(p,q):=C(p,q)-c_{14}p^{14}-c_0q^{14}/pq; \quad (7.2)$$

$\{$modified values of $C(p,q)$, see $(2.5)\}$

In order to separate "odd" and "even" unknowns, compute

$$\begin{aligned}S(1,1)&:=[M(1,1)+M(1,-1)]/2\\ &=c_{12}+c_{10}+c_8+c_6+c_4+c_2;\\ S(2,1)&:=[M((2,1))+M((2,-1)]/8\\ &=2^{10}c_{12}+2^8c_{10}+..+c_2.\end{aligned} \quad (7.3)$$

In general, by computing

$$S(p,q):=[M(p,q)+M(p,-q)]/2(pq)^2; \quad (7.4)$$

for $(p,q)=\{(1,1);(2,1);(1,2);(3,1);(1,3);(3,2)\}$ we create *six* equations with *six* "even" unknowns $c_2,c_4,\cdots,c_{12}$.

Analogously, by computing

$$D(p,q):=[M(p,q)-M(p,-q)]/2; \quad (7.5)$$

for $(p,q)=\{(1,1);(2,1);(1,2);(3,1);(1,3);(3,2)\}$, we create *six* equations with *seven* "odd" unknowns $c_1,c_3,\cdots,c_{13}$.

After the values of "even" coefficients $c_2,c_4,\cdots,c_{12}$ are computed, we use $M(2,3)$ {see (7.2)} as the *seventh* equation for computation of all "odd" coefficients.

## 8. AHP for *n* = 4 in Details

Consider

$$C(x,y):=A(x,y)B(x,y); \quad (8.1)$$

where

$$A(x,y):=\sum_{k=0}^{3}a_kx^ky^{3-k}; \quad (8.2)$$

$$B(x,y):=\sum_{k=0}^{3}b_kx^ky^{3-k}; \quad (8.3)$$

and

$$C(x,y):=\sum_{k=0}^{6}c_ix^iy^{6-i}; \quad (8.4)$$

then

$$C(0,1)=c_0=a_0b_0;\ \ C(1,0)=c_6=a_3b_3; \quad (8.5)$$

$$\begin{aligned}C(1,1)&:=c_6+c_5+\cdots+c_1+c_0\\ &=(a_3+\cdots+a_0)(b_3+\cdots+b_0);\end{aligned} \quad (8.6)$$

$$\begin{aligned}C(1,-1)&:=c_6-c_5+\cdots+c_0\\ &=(-a_3+\cdots+a_0)(-b_3+\cdots+b_0);\end{aligned} \quad (8.7)$$

$$\begin{aligned}C(2,1)&:=64c_6+32c_5+16c_4+\cdots+2c_1+c_0\\ &=(8a_3+4a_2+2a_1+a_0)(8b_3+4b_2+2b_1+b_0);\end{aligned} \quad (8.8)$$

$$\begin{aligned}C(2,-1)&:=64c_6-32c_5+\cdots-2c_1+c_0\\ &=(-8a_3+4a_2-2a_1+a_0)(-8b_3+4b_2-2b_1+b_0);\end{aligned} \quad (8.9)$$

$$\begin{aligned}C(1,2)&:=c_6+2c_5+4c_4+\cdots+32c_1+64c_0\\ &=(a_3+2a_2+4a_1+8a_0)(b_3+2b_2+4b_1+8b_0).\end{aligned} \quad (8.10)$$

## 9. Solution of System of Equations (8.6)-(8.10)

**Step 9.1:** $V_1:=C(1,1)-c_6-c_0$;

**Step 9.2:** $V_2:=c_6+c_0-C(1,-1)$;

**Step 9.3:** $V_3:=[C(2,1)-c_0]/2-32c_6$;

**Step 9.4:** $V_4:=32c_6+[c_0-C(2,-1)]/2$;

**Step 9.5:** $V_5:=[C(1,2)-c_6]/2-32c_0$.

***Remark*** **9.1:** Using $V_1$-$V_5$, we find five unknowns $c_1,\cdots,c_5$ from five linear equations:

$$c_5+c_4+c_3+c_2+c_1=V_1; \quad (9.1)$$

$$c_5 - c_4 + c_3 - c_2 + c_1 = V_2; \quad (9.2)$$

$$16c_5 + 8c_4 + 4c_3 + 2c_2 + c_1 = V_3; \quad (9.3)$$

$$16c_5 - 8c_4 + 4c_3 - 2c_2 + c_1 = V_4; \quad (9.4)$$

$$c_5 + 2c_4 + 4c_3 + 8c_2 + 16c_1 = V_5. \quad (9.5)$$

**Step 9.6:**

$$V_6 := (V_1 - V_2)/2 \ \{= c_4 + c_2\}; \quad (9.6)$$

**Step 9.7:**

$$V_7 := (V_3 - V_4)/4 \ \{= 4c_4 + c_2\};. \quad (9.7)$$

**Step 9.8:**

$$c_4 := (V_7 - V_6)/3; \quad (9.8)$$

**Step 9.9:**

$$c_2 := V_6 - c_4. \quad (9.9)$$

**Step 9.10:**

$$B_1 := V_1 - c_2 - c_4; \ B_2 := V_3 - 8c_4 - 2c_2;$$
$$B_3 := V_5 - 2c_4 - 8c_2.$$

***Remark* 9.2:** Now we solve the system of three equations with three unknowns:

$$c_5 + c_3 + c_1 = B_1; \quad (9.10)$$

$$16c_5 + 4c_3 + c_1 = B_2; \quad (9.11)$$

$$c_5 + 4c_3 + 16c_1 = B_3. \quad (9.12)$$

**Step 9.11:**

$$B_4 := (B_2 - 4B_1)/3 \ \{= 4c_5 - c_1\}; \quad (9.13)$$

**Step 9.12:**

$$B_5 := (B_3 - 4B_1)/3 \ \{= -c_5 + 4c_1\}. \quad (9.14)$$

**Step 9.13:**

$$B_6 := (B_4 + B_5)/3 \ \{= c_5 + c_1\}; \quad (9.15)$$

**Step 9.14:**

$$c_1 := (B_5 + B_6)/5; \ c_5 := B_6 - c_1; \quad (9.16)$$

**Step 9.15:**

$$c_3 = B_1 - c_5 - c_1. \quad (9.17)$$

## 10. Multistage Implementation of TCA and AHP

### 10.1. Two-Stage Implementation (TSI)

Let us consider $n = 6 = 2 \times 3$, and analyze how to multiply sextuple-large integers in two stages. On the first stage, we represent $A$ and $B$ as double long:

$$A = 10^{3m} A_1 + A_0; \ B = 10^{3m} B_1 + B_0; \quad (10.1)$$

and compute $AB$ applying the Karatsuba algorithm [1], which requires *three* multiplications of 3*m*-long integers.

Then, as the second stage, we compute every product of triple-long integers using either the TCA or AHP each requiring *five* standard multiplications {SMs, for short}. Therefore, the two-stage implementation requires *fifteen* SMs rather than *eleven* SMs required by the TCA or by AHP. **Table 2** provides comparison for several other cases, where

$$R(n) := TSI(n)/DI(n). \quad (10.2)$$

***General case*:** Let us now analyze the two-stage implementation of a multiplication algorithm if $n = rs$.

First, we represent $A$ and $B$ as

$$A = \sum_{k=0}^{r-1} 10^{ksm} A_k; \quad (10.3)$$

and

$$B = \sum_{k=0}^{r-1} 10^{ksm} B_k; \quad (10.4)$$

Such an implementation requires $2r - 1$ multiplications each of *sm*-long integers. Then, on the second stage, we multiply *sm*-long integers. Every such multiplica-tion requires $2s - 1$ SMs. Therefore, we need

$$(2r-1)(2s-1) = O(4n) \quad (10.5)$$

SMs in total. However, the direct (one-stage) multiplication requires only

$$D(n) = 2n - 1 = O(2n) \text{ SMs}. \quad (10.6)$$

It is easy to verify that both parts in the inequality

$$(2r-1)(2s-1) \geq 2rs - 1 \quad (10.7)$$

are equal if and only if either $r = 1$, or $s = 1$, or $r = s = 1$. In all other cases

**Table 2. Number of SMs in two-stage (TSI) and direct implementation (DI).**

| ***n*** | **4** | **6** | **9** | **15** | **21** | **25** | **35** | **49** | **121** |
|---|---|---|---|---|---|---|---|---|---|
| ***TSI*** | 9 | 15 | 25 | 45 | 65 | 81 | 117 | 169 | 441 |
| ***DI*** | 7 | 11 | 17 | 29 | 41 | 49 | 69 | 97 | 241 |
| ***R*(*n*)** | 1.29 | 1.36 | 1.47 | 1.55 | 1.59 | 1.65 | 1.70 | 1.74 | 1.83 |

$$(2r-1)(2s-1) > 2rs-1. \qquad (10.8)$$

Thus, the TSI of either the TCA or AHP for large $r$ and $s$ asymptotically requires twice as many SMs than the DI.

***Example* 10.1:** Now let $m$ = 4;

$$A = 385,425,374,179;$$

and

$$B = 608,368,695,784.$$

Therefore, in this example we need to split $A$ and $B$ into three parts, *i.e.*, $n$ = 3.

By the algorithm, described in Sections 8 and 9 we can compute $C = AB$ using five multiplications of four-digit large integers. However, if the standard integers are only two-digit long, we can pre-compute each product recursively using the Karatsuba algorithm. Each of these five products requires three SMs, *i.e.*, overall we need *fifteen* SMs to compute $C = AB$. On the other hand, we can compute the same product $AB$ splitting both $A$ and $B$ into *six* parts.

In this case $m$ = 2. To compute $AB$, we need only *eleven* SMs by using the direct implementation vs *fifteen* SMs required in the recursive TSI implementation.

### 10.2. Multi-Stage Implementation

Let now $n = 8 = 2^3$. Therefore, we multiply $A$ and $B$ in either three stages using the Karatsuba algorithm [1] or using the AHP or TCA directly. In the MSI we need $3^3 = 27$ SMs; while in the DI we need only *fifteen* SMs.

***Remark* 10.1:** Since the number of algebraic additions in the DI asymptotically grows as function of $n$, it is essential to properly select the evaluation points $(p,q)$ to implement symmetricity illustrated above in Section 4 {see (4.4)-4.16)} and to simplify computational complexity stemming from the multiplication by constant coefficients. These issues are addressed in Sections 11-13.

## 11. Number of Algebraic Additions

Notice that computation of $M(p,q)$ requires $2n$ additions of SIs. Since we need to compute $M(p,q)$ for $2n-3$ different values of $(p,q)$, the total number of algebraic additions is of order $O(4n^2)$. This number can be reduced twice as demonstrated in Section 2. Since every addition of $m$-digit long integers has order $O(m)$, therefore the total complexity of all additions is of order $O(2mn^2)$. Hence, the overall complexity is equal

$$T(m,n) = O(2nm^2 + 2mn^2) = O[2nm(m+n)].$$
$$\text{If } m \gg n, \text{ then } T(m,n) = O(2nm^2). \qquad (11.1)$$

## 12. Analysis of TCA vs AHP

In large-integer multiplication we addressed two sources of complexity: the number of standard multiplications and the number of algebraic additions. The third source of complexity is multiplication by constant coefficients when the polynomials $A(x,y)$ and $B(x,y)$ are evaluated at points $(x,y)=(p,q)$.

The **Table 3** compares the polynomial evaluations in the TCA and AHP frameworks respectively for various values of $n$. It means that if $n$ = 15, then in TCA polynomials $C(p)$ are evaluated for $p = \{0,\pm1,\pm2,\cdots,\pm14\}$ and in the corresponding AHP polynomials $C(p,q)$ are evaluated for

$$(p,q) = \{(0,1),(1,\pm1),(2,\pm1),(1,\pm2),\cdots,$$
$$(1,\pm5),(5,\pm2),(2,\pm5),(5,\pm4),(4,\pm5)\}$$

***Example* 12.1:** Compare for $n$ = 14 the computation of $C$(2,5) and $C$(13):

$$C(13) = \sum_{k=0}^{13} 13^k a_k \times \sum_{k=0}^{13} 13^k b_k;$$
$$C(5,2) = \sum_{k=0}^{13} 2^k \times 5^{13-k} a_k \times \sum_{k=0}^{13} 2^k \times 5^{13-k} b_k. \qquad (12.1)$$

In the next section we provide an iterative procedure that computes $C(p,q)$.

## 13. Generalized Horner Rule for Homogeneous Polynomial

Let $R_0 := a_0$; $L_0 := a_n$; and for $k = 1,\cdots,n$

**Table 3. Evaluations required in Toom's vs AHP frameworks.**

| ***n*** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **TCA** | $C(0)$ | $C(\pm1)$ | $C(\pm2)$ | $C(\pm3)$ | $C(\pm4)$ | $C(\pm5)$ | $C(\pm7)$ | $C(\pm7)$ |
| **AHP** | $C(0,1)$ | $C(1,\pm1)$ | $C(2,\pm1)$ | $C(1,\pm2)$ | $C(3,\pm1)$ | $C(1,\pm3)$ | $C(3,\pm2)$ | $C(2,\pm3)$ |
| ***n*** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **TCA** | $C(\pm8)$ | $C(\pm9)$ | $C(\pm10)$ | $C(\pm11)$ | $C(\pm12)$ | $C(\pm13)$ | $C(\pm14)$ | $C(\pm15)$ |
| **AHP** | $C(4,\pm1)$ | $C(1,\pm4)$ | $C(5,\pm1)$ | $C(1,\pm5)$ | $C(5,\pm2)$ | $C(2,\pm5)$ | $C(5,\pm4)$ | $C(4,\pm5)$ |

$$R_k := R_{k-1}q + a_k p^k; \quad (13.1)$$

and

$$L_k := L_{k-1}p + a_{n-k}q^k; \quad (13.2)$$

then

$$A(p,q) = L_n = R_n. \quad (13.3)$$

Analogously, we can compute $B(p,q)$.

## 14. Values of $(p,q)$ Simplifying Computation of $A(p,q)$ and $B(p,q)$

**Case 1:** if $p = 2^s$ and $q = 2^t \pm 1$ in (13.1), then $a_k p^k$ requires a binary shift on *sk* positions, and $R_{k-1}q$ requires merely a binary shift on *t* positions and one algebraic addition [3].

**Case 2:** if $q = 2^s$ and $p = 2^t \pm 1$ in (13.2), then, analogously as in the Case1, $a_{n-k}q^k$ requires a binary shift on *sk* positions, and $L_{k-1}p$ requires a binary shift on *t* positions and one algebraic addition.

**Case 3:** if $q = 2^r(2^t \pm 1)$ and $p = 1$ in (13.1), then it is necessary to use two binary shifts and one algebraic addition.

**Case 4:** if $p = 2^r(2^t \pm 1)$ and $q = 1$ in (13.2), then, analogously as in the Case 3, it is necessary to use two binary shifts and one algebraic addition.

**Case 5:** if $q = 2^r$ and $p = 1$ in (13.1), then it is necessary to use only one binary shift on *r* positions.

**Case 6:** if $p = 2^r$ and $q = 1$ in (13.2), then, analogously as in the Case 5, it is necessary to use only one binary shift.

***Example* 14.1:** In the following set of 37 points each $(p,q)$ satisfies one of six special cases listed above:

$$\begin{aligned}(p,q) \in \{&(1,1);(2,1);(3,1);\cdots;(10,1);\\ &(12,1);(14,1);\cdots;(18,1);(20,1);(24,1);\\ &(28,1);(30,1);(31,1);\\ &(3,2);(5,2);(7,2);(9,2);(15,2);\\ &(17,2);(31,2);\\ &(5,4);(7,4);(9,4);(15,4);(17,4);\\ &(9,8);(15,8);(17,8);\ \text{and}\ (17,16)\}.\end{aligned}$$

Add to this set the other 111 points $(p,-q)$, $(q,p)$ and $(q,-p)$. For each of these 148 points the number of required algebraic additions in the AHPs is smaller than in the corresponding TCAs.

***Example* 14.2:** If *n* = 22, then for the TCA we need to evaluate $C(p)$ at 43 points $C(0)$, $C(\pm1),\cdots,C(\pm12)$; yet, for the AHP we evaluate polynomial $C(p,q)$ at points

$$\begin{aligned}&C(0,1);C(1,0);\\ &C(1,\pm1);C(2,\pm1);\cdots;C(7,\pm1);\\ &C(1,\pm2);C(1,\pm3);\cdots;C(1,\pm7);\\ &C(3,\pm2);C(5,\pm2);C(7,\pm2);\\ &C(2,\pm3);C(2,\pm5);C(2,\pm7);\end{aligned}$$

and $C(5,4)$, where, for instance, the evaluation of *C*(5,4) requires fewer basic operations than for *C*(21) in the TCA.

## 15. Optimized AHP

In order to decrease twice the number of additions/subtractions, we need to adjust the Generalized Horner Rule for iterative computation of $A(x,y)$ and $B(x,y)$.

Notice that if *n* is *odd* $\{n = 2s-1\}$, then

$$\begin{aligned}&A_{2s-1}(p,q)\\ &=\left(a_{2(s-1)}p^{2(s-1)} + \cdots + a_2p^2q^{2(s-2)} + a_0q^{2(s-1)}\right)\\ &\quad+\left(a_{2s-3}p^{2s-3}q + \cdots + a_1pq^{2s-3}\right)\\ &=A^{(0)}_{2s-1}(p,q) + A^{(1)}_{2s-1}(p,q).\end{aligned} \quad (15.1)$$

Otherwise, if *n* is *even*, *i.e.*, if *n* = 2*s*, then

$$\begin{aligned}&A_{2s}(p,q)\\ &=\left(a_{2s-1}p^{2s-1} + a_{2s-3}p^{2s-3}q^2 + \cdots + a_3p^3q^{2(s-2)} + a_1pq^{2(s-1)}\right)\\ &\quad+\left(a_{2(s-1)}p^{2(s-1)}q + \ldots + a_2p^2q^{2s-3} + a_0q^{2s-1}\right)\\ &= A^{(0)}_{2s}(p,q) + A^{(1)}_{2s}(p,q).\end{aligned} \quad (15.2)$$

Therefore, for every even and odd *n*

$$A_n(p,q) = A^{(0)}_n(p,q) + A^{(1)}_n(p,q); \quad (15.3)$$

$$A_n(p,-q) = A^{(0)}_n(p,q) - A^{(1)}_n(p,q). \quad (15.4)$$

Let us show how to modify (13.1) for iterative computation of (15.1).

Consider $n = 2s-1$; $d_i := a_{n-1-i}$; assign $R^{(0)}_0 := a_0$; $L^{(0)}_0 := d_0$; and for every $k = 1,2,\cdots$ compute

$$R^{(0)}_{2k} := R^{(0)}_{2(k-1)}q^2 + a_{2k}p^{2k}; \quad (15.5)$$

$$L^{(0)}_{2k} := L^{(0)}_{2(k-1)}p^2 + d_{2k}q^{2k}; \quad (15.6)$$

hence,

$$A^{(0)}_{2s-1}(p,q) = R^{(0)}_{2(s-1)} = L^{(0)}_{2(s-1)}. \quad (15.7)$$

Assign $R^{(1)}_0 := 0$; $L^{(1)}_0 := 0$;
and for every $k = 1,2,\cdots$ compute

$$\begin{aligned}R^{(1)}_{2k} &= R^{(1)}_{2(k-1)}q^2 + a_{2k-1}p^{2k-1}q;\\ L^{(1)}_{2k} &= L^{(1)}_{2(k-1)}p^2 + d_{2k-1}q^{2k-1}p.\end{aligned} \quad (15.8)$$

hence

$$A^{(1)}_{2s-1}(p,q) = R^{(1)}_{2(s-1)} = L^{(1)}_{2(s-1)}.$$

Thus,

$$A_{2s-1}(p,q) = R^{(0)}_{2(s-1)} + R^{(1)}_{2(s-1)} = L^{(0)}_{2(s-1)} + L^{(1)}_{2(s-1)}; \quad (15.9)$$

$$A_{2s-1}(p,-q) = R^{(0)}_{2(s-1)} - R^{(1)}_{2(s-1)}. \quad (15.10)$$

***Example* 15.1:** Let us consider *n* = 7. Then

$$\begin{aligned} A_7(p,q) &= \left(a_6p^6 + a_4p^4q^2 + a_2p^2q^4 + a_0q^6\right) \\ &\quad + \left(a_5p^5q + a_3p^3q^3 + a_1pq^5\right) \\ &= R^{(0)}_6 + R^{(1)}_6. \end{aligned}$$

The iterative procedures (15.5)-(15.8) are simplified if
1) *p* is a power of 2 and $q = 2^t \pm 1; t \geq 1$ or
2) *p* = 1 and $q = 2^r\left(2^t \pm 1\right)$; {see the corresponding Cases 1-6 in Section 13}.

The Equations (13.2), (15.5) and (15.6) can be analogously modified for iterative computation of

$$A^{(0)}_{2s}(p,q); A^{(1)}_{2s}(p,q);$$

$$B^{(0)}_{2s-1}(p,q); B^{(1)}_{2s-1}(p,q);$$

and

$$B^{(0)}_{2s}(p,q); B^{(1)}_{2s}(p,q).$$

## 16. Conclusion

It has been demonstrated that the overhead in the Toom-Cook algorithm is higher than in the proposed approach based on homogeneous polynomials $A(x,y)$ and $B(x,y)$. Integrality of all coefficients in the TCA and AHP is demonstrated by the first author in [11].

## 17. Acknowledgements

I deeply appreciate comments of I. V. Semushin and stylistic suggestions of Yu. Polyakov and R. Rubino that improved the quality of this paper. I wish to express my gratitude to typesetters for their patience.

## REFERENCES

[1] A. Karatsuba and Yu. Ofman, "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady*, Vol. 7, 1963, pp. 595-596.

[2] A. Toom, "The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers," *Soviet Mathematics-Doklady*, Vol. 7, 1963, pp. 714-716.

[3] S. A. Cook, "On the Minimum Computation Time of Functions," Chapter 3, Ph.D. Thesis, Harvard University, Cambridge, 1966, pp. 51-77.

[4] D. Knuth, "Art of Computer Programming: Seminumerical Algorithms," 2nd Edition, Vol. 2, Addison-Wesley, New York, 1981.

[5] R. Crandall and C. Pomerance, "Prime Numbers: A Computational Perspective," Springer, New York, 2001. doi:10.1007/978-1-4684-9316-0

[6] D. Bernstein, "Multidigit Modular Multiplication with Explicit Chinese Remainder Theorem," 1997. ftp://koobera. math.uic.edu/pub/papers/m3.dvi

[7] R. Crandall, "Method and Apparatus for Public Key Exchange in a Cryptographic Systems," US Patents 5159632, 1992; 5271061, 1993; 5463690, 1994.

[8] F. Ablayev and M. Karpinski, "A Lower Bound for Integer Multiplication on Randomized Ordered Read-Once Branching Programs," *Information and Computation*, Vol. 186, No. 1, 2003, pp. 78-89. doi:10.1016/S0890-5401(03)00118-4

[9] A. Zanoni, "Iterative Toom-Cook Methods for Very Unbalanced Long Integer Multiplication," *Proceedings of the* 2010*th International Symposium on Symbolic and Algebraic Computation*, Munich, 25 July 2010, pp. 319-323. doi:10.1145/1837934.1837995

[10] W. G. Horner, "A New Method of Solving Numerical Equations of All Orders, by Continuous Approximation," *Philosophical Transactions of Royal Society of London*, Vol. 109, 1819, pp. 308-335. doi:10.1098/rstl.1819.0023

[11] B. Verkhovsky and R. Rubino, "Corporate Intranet Security: Packet-Level Protocols for Preventing Leakage of Sensitive Information and Assuring Authorized Network Traffic," *International Journal of Communications, Network and System Sciences*, Vol. 5, No. 5, 2012, pp. 517-524.