A Novel Method of Modular Multiplication Based on Karatsuba-like Multiplication

Zhen Gu
Institute of Microelectronics
Tsinghua University
Beijing, China
guz15@mails.tsinghua.edu.cn

Shuguo Li
Institute of Microelectronics
Tsinghua University
Beijing, China
lisg@tsinghua.edu.cn

***Abstract*—In this paper, we propose a novel method of modular multiplication which embeds the modular reduction in the evaluation and interpolation parts of the Karatsuba-like multiplication. Before, the modular reduction can only be performed independently between multiplication. However, applying our method, the interpolation of the previous multiplication, modular reduction and evaluation of the next multiplication are merged as a whole step, which leads to the simplification of computations and improvement of parallelism. This method can be applied to the modular multiplication with simple moduli like NIST primes, and for general moduli, we can apply this method by using Montgomery modular multiplication instead.**

***Index Terms*—Modular multiplication, Karatsuba-like multiplication, Montgomery modular multiplication**

I. Introduction

Modern public-key cryptosystems are mostly based on the finite-field arithmetic. [1] [2] [3] The finite-field arithmetic usually involves various modular additions and modular multiplications, where modular additions are simple in implementation and fast in speed while modular multiplications are much slower and more complex. In this sense, the design and speed-up of modular multiplications are significant to the implementation of efficient public-key cryptosystems. Basically, the speed-up of modular multiplications mainly depends on that of integer multiplications. Integer multiplications can thus be speed up using multiplication algorithms, like Karatsuba multiplication, Toom-Cook multiplication, FFT-based multiplication, and many others. [4] [5] [6] [7] [8] In applications of modular multiplications, Karatsuba multiplication is quite often applied since it is simple for implementation and great in performance for applications nowadays. Karatsuba-like multiplications are developed to split the integers into even smaller sizes and maintain the approximate complexity of Karatsuba multiplication. To shorten the steps between integer multiplications, we propose a novel method for modular multiplication based on Karatsuba-like multiplications.

The authors are with Institute of Microelectronics, Tsinghua University, Beijing 100084, China. This work was supported by the National Natural Science Foundation of China under Grant 61674086 and 61974083.

II. Karatsuba-like Multiplication

A. Karatsuba Multiplication

Karatsuba multiplication is based on the following observation,

$$\begin{aligned}&(a_1x+a_0)(b_1x+b_0)\\=&a_1b_1x^2+(a_1b_0+a_0b_1)x+a_0b_0\\=&a_1b_1x^2+[(a_1+a_0)(b_1+b_0)-a_1b_1-a_0b_0]x+a_0b_0,\end{aligned}$$

where the multiplication of two linear polynomials requires 3 multiplications $a_1b_1, a_0b_0, (a_1+a_0)(b_1+b_0)$ rather than 4 multiplications $a_1b_1, a_1b_0, a_0b_1, a_0b_0$ as required when Karatsuba multiplication is not applied. [4] [8] [9] [10] [11]

Before further discussions, we introduce some useful notations for the simplification of Karatsuba-like multiplication.

1) Base Word β: An integer β chosen as a parameter.
2) Power Vector P_n: A power vector of length n is a vector of consecutive increasing powers of β, for example, $P_n^T = \left(\begin{smallmatrix} 1 & \beta & \beta^2 & \cdots & \beta^{n-1}\end{smallmatrix}\right)$
3) Coefficient Vector $A(B)$: For an integer $a = \sum_{i=0}^{n-1} a_i\beta^i$ of n words, its coefficient vector then is $A^T = \left(\begin{smallmatrix} a_0 & a_1 & \cdots & a_{n-1}\end{smallmatrix}\right)$.
4) Evaluation Vector $\hat{A}(\hat{B})$: A vector $\hat{A}$ of linear combinations of elements in its corresponding coefficient vector A.
5) Evaluation Matrix E: A matrix of constant entries to transform coefficient vectors to evaluation vectors when multiplied by the evaluation matrix E.
6) Interpolation Matrix I: A matrix of constant entries to transform evaluation vectors to coefficient vectors when multiplied by the interpolation matrix I.

For Karatsuba multiplication, if we have

$$a = a_1\beta + a_0$$
$$b = b_1\beta + b_0$$

Then the corresponding coefficient vectors are

$$A = \left(\begin{smallmatrix} a_0 \\ a_1\end{smallmatrix}\right)$$
$$B = \left(\begin{smallmatrix} b_0 \\ b_1\end{smallmatrix}\right).$$

Moreover, it is obvious that $P_2^T \cdot A = a,\ P_2^T \cdot B = b$. The evaluation matrix is

$$E = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1\end{smallmatrix}\right).$$

Hence

$$\hat{A} = E \cdot A = \begin{pmatrix} a_0 \\ a_0+a_1 \\ a_1 \end{pmatrix}$$
$$\hat{B} = E \cdot B = \begin{pmatrix} b_0 \\ b_0+b_1 \\ b_1 \end{pmatrix}.$$

Afterwards, the point-wise product ($\otimes$) $\hat{C}$ of vectors $\hat{A}, \hat{B}$ is

$$\hat{C} = \hat{A} \otimes \hat{B} = \begin{pmatrix} a_0 b_0 \\ (a_0+a_1)(b_0+b_1) \\ a_1 b_1 \end{pmatrix}.$$

The interpolation matrix is

$$I = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore, the evaluation vector C is

$$C = I \cdot \hat{C} = \begin{pmatrix} a_0 b_0 \\ (a_0+a_1)(b_0+b_1) - a_0 b_0 - a_1 b_1 \\ a_1 b_1 \end{pmatrix}.$$

Thus the overall product c can be obtained as

$$c = P_3^T \cdot C.$$

Evaluation and interpolation is usually used in interpolation-based multiplications like Karatsuba multiplication, Toom-Cook multiplication and FFT-based multiplication. [4] [5] [6] [8] [7] However, in this paper, we still refer the steps as evaluation and interpolation when we are dealing with Karatsuba-like multiplications.

B. Karatsuba-like Multiplication

Inspired by Karatsuba multiplication, which computes the product using the substitution $a_1b_0 + a_0b_1 = (a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1$, researchers have developed Karatsuba-like multiplication which does not restrict the word-length to be 2. [12] [13] [14] [15] [16] In Karatsuba-like multiplications, for example, n-Term Karatsuba multiplication, it can be computed in at most $M(n)$ base-word multiplications. A list of $M(n)$ in [12] can be shown in Table I. We present a 3-Term Karatsuba multiplication here as described in [12],

TABLE I
NUMBER OF BASE-WORD MULTIPLICATIONS $M(n)$ FOR n-TERM KARATSUBA MULTIPLICATION IN [12]

n	$M(n)$	n	$M(n)$
1	1	2	3
3	6	4	9
5	13	6	17
7	22	8	27
9	34	10	39

$$\begin{aligned}
&(a_0 + a_1x + a_2x^2)(b_0 + b_1x + b_2x^2) \\
=&a_0b_0(1 - x) \\
&+a_1b_1(-x + 2x^2 - x^3) \\
&+a_2b_2(-x^3 + x^4) \\
&+(a_0 + a_1)(b_0 + b_1)(x - x^2) \\
&+(a_1 + a_2)(b_1 + b_2)(-x^2 + x^3) \\
&+(a_0 + a_1 + a_2)(b_0 + b_1 + b_2)x^2
\end{aligned}$$

Hence, we have its evaluation matrix E can then be derived, which is,

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

, and correspondingly, its interpolation matrix is

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & -1 & -1 & 1 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

To summarize, the parameters for n-term Karatsuba-like multiplications are

1) n-dim single-word coefficient vectors A, B
2) $M(n) \times n$ evaluation matrix E
3) $M(n)$-dim single-word evaluation vectors $\hat{A}, \hat{B}$
4) $M(n)$-dim double-word evaluation vector $\hat{C}$
5) $(2n - 1) \times M(n)$ interpolation matrix I
6) $(2n - 1)$-dim power vector P_{2n-1}.

In 3-term Karatsuba-like multiplications described above, $n = 3$ and $M(n) = 6$. Correspondingly, 3-term Karatsuba multiplications have parameters E being 6×3, I being 5×6. We further present here another example of 4-term Karatsuba multiplications as both a representative of Karatsuba-like multiplications and an example of recursive usage of Karatsuba-like multiplications, which can be viewed as two 2-term Karatsuba multiplications are recursively applied.

We firstly have a look into the 4-term Karatsuba multiplication similarly to the 3-term Karatsuba multiplication described above. For two cubic polynomials, we have

$$\begin{aligned}
&(a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3) \\
=&a_0b_0(1 - x - x^2 + x^3) \\
&+a_1b_1(-x + x^2 + x^3 - x^4) \\
&+a_2b_2(-x^2 + x^3 + x^4 - x^5) \\
&+a_3b_3(x^3 - x^4 - x^5 + x^6) \\
&+(a_0 + a_1)(b_0 + b_1)(x - x^3) \\
&+(a_0 + a_2)(b_0 + b_2)(x^2 - x^3) \\
&+(a_1 + a_3)(b_1 + b_3)(-x^3 + x^4) \\
&+(a_2 + a_3)(b_2 + b_3)(-x^3 + x^5) \\
&+(a_0 + a_1 + a_2 + a_3)(b_0 + b_1 + b_2 + b_3)x^3
\end{aligned}$$

Hence, the evaluation matrix E is,

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

and the interpolation matrix I is,

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 \\ 0 & -1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since $M(4) = M(2) \times M(2)$, we can applied 2-term Karatsuba multiplications recursively to obtain another form of 4-term Karatsuba multiplication. For two cubic polynomials, let $y = x^2$, then

$$
\begin{aligned}
&(a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3) \\
=&((a_0 + a_1x) + (a_2 + a_3x)y)((b_0 + b_1x) + (b_2 + b_3x)y) \\
=&(a_0 + a_1x)(b_0 + b_1x) \\
&+[((a_0 + a_2) + (a_1 + a_3)x)((b_0 + b_2) + (b_1 + b_3)x) \\
&\quad - (a_0 + a_1x)(b_0 + b_1x) - (a_2 + a_3x)(b_2 + b_3x)]y \\
&+(a_2 + a_3x)(b_2 + b_3x)y^2 \\
=&\begin{pmatrix} 1-y \\ y \\ y^2-y \end{pmatrix}^T \cdot \left[\begin{pmatrix} a_0+a_1x \\ (a_0+a_2)+(a_1+a_3)x \\ a_2+a_3x \end{pmatrix} \otimes \begin{pmatrix} b_0+b_1x \\ (b_0+b_2)+(b_1+b_3)x \\ b_2+b_3x \end{pmatrix} \right]
\end{aligned}
$$

Hence, the product can be viewed as product of linear polynomials in y, where the product can be derived via three multiplications of linear polynomials in x each when the Karatsuba multiplication is applied. Then for each multiplication of linear polynomials in x, the product can be further obtained by applying another Karatsuba multiplication where we take three multiplications of integers. Thus, the multiplication of two cubic polynomials can be derived via nine multiplications of integers. The following formulas present how the nine multiplications are obtained.

$$
\begin{aligned}
&(a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3) \\
=&\begin{pmatrix} 1-y \\ y \\ y^2-y \end{pmatrix}^T \cdot \left[\begin{pmatrix} a_0+a_1x \\ (a_0+a_2)+(a_1+a_3)x \\ a_2+a_3x \end{pmatrix} \otimes \begin{pmatrix} b_0+b_1x \\ (b_0+b_2)+(b_1+b_3)x \\ b_2+b_3x \end{pmatrix} \right] \\
=&\begin{pmatrix} 1-y \\ y \\ y^2-y \end{pmatrix}^T \cdot \begin{pmatrix} \begin{pmatrix} 1-x \\ x \\ x^2-x \end{pmatrix}^T \cdot \begin{pmatrix} a_0b_0 \\ (a_0+a_1)(b_0+b_1) \\ a_1b_1 \end{pmatrix} \\ \begin{pmatrix} 1-x \\ x \\ x^2-x \end{pmatrix}^T \cdot \begin{pmatrix} (a_0+a_2)(b_0+b_2) \\ \sum_{i=0}^3 a_i \sum_{i=0}^3 b_i \\ (a_1+a_3)(b_1+b_3) \end{pmatrix} \\ \begin{pmatrix} 1-x \\ x \\ x^2-x \end{pmatrix}^T \cdot \begin{pmatrix} a_2b_2 \\ (a_2+a_3)(b_2+b_3) \\ a_3b_3 \end{pmatrix} \end{pmatrix} \\
=&\begin{pmatrix} (1-y)(1-x) \\ (1-y)x \\ (1-y)(x^2-x) \\ y(1-x) \\ yx \\ y(x^2-x) \\ (y^2-y)(1-x) \\ (y^2-y)x \\ (y^2-y)(x^2-x) \end{pmatrix}^T \cdot \begin{pmatrix} a_0b_0 \\ (a_0+a_1)(b_0+b_1) \\ a_1b_1 \\ (a_0+a_2)(b_0+b_2) \\ \sum_{i=0}^3 a_i \sum_{i=0}^3 b_i \\ (a_1+a_3)(b_1+b_3) \\ a_2b_2 \\ (a_2+a_3)(b_2+b_3) \\ a_3b_3 \end{pmatrix} \\
=&\begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0b_0 \\ (a_0+a_1)(b_0+b_1) \\ a_1b_1 \\ (a_0+a_2)(b_0+b_2) \\ \sum_{i=0}^3 a_i \sum_{i=0}^3 b_i \\ (a_1+a_3)(b_1+b_3) \\ a_2b_2 \\ (a_2+a_3)(b_2+b_3) \\ a_3b_3 \end{pmatrix}
\end{aligned}
$$

where the last step is obtained by substituting y with x^2 in the expressions. Therefore, we have obtained another valid I and corresponding E for 4-term Karatsuba multiplications, which are

$$
I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

As we can see, I's derived from two different methods mentioned above are identical in the sense of column interchanges. Correspondingly, the E's are identical in the sense of row interchanges. To summarize, for $n = n_1 \cdots n_k$, one has

$$
M(n) = \prod_{i=1}^{k} M(n_i)
$$

Then, the parameters of n-term Karatsuba multiplications can be derived from the k-level recursive applications of Karatsuba-like multiplications similarly to 4-term Karatsuba multiplications.

We perform an example and it would go through the whole paper in the following discussions. We take

$$
\beta = 10^2 \ a = 12345678 \ b = 21436587 \tag{1}
$$

Then

$$
\begin{aligned}
&A = \\
&\begin{pmatrix} 78 & 56 & 34 & 12 \end{pmatrix}^T \\
&B = \\
&\begin{pmatrix} 87 & 65 & 43 & 21 \end{pmatrix}^T \\
&\hat{A} = E \cdot A = \\
&\begin{pmatrix} 78 & 56 & 34 & 12 & 134 & 112 & 68 & 46 & 180 \end{pmatrix}^T \\
&\hat{B} = E \cdot B = \\
&\begin{pmatrix} 87 & 65 & 43 & 21 & 152 & 130 & 86 & 64 & 216 \end{pmatrix}^T \\
&\hat{C} = (E \cdot A) \otimes (E \cdot B) = \\
&\begin{pmatrix} 6786 & 3640 & 1462 & 252 & 20368 & 14560 & 5848 & 2944 & 38880 \end{pmatrix}^T \\
&C = I \cdot ((E \cdot A) \otimes (E \cdot B)) = \\
&\begin{pmatrix} 6786 & 9942 & 9952 & 7300 & 3418 & 1230 & 252 \end{pmatrix}^T \\
&c = P_7^T \cdot (I \cdot ((E \cdot A) \otimes (E \cdot B))) = \\
&\ 264649200520986 \\
&c = a \cdot b = 12345678 \times 21436587 \\
&\ 264649200520986
\end{aligned}
$$

As can be seen from the above example, we can compute the product of two integers with 9 integer multiplication operations and other computations which are related to the multiplication by evaluation matrices whose every entry is either 0 or ± 1. Also, in this example, we let the base word be 10^2 rather than a power of 2 in order to have a more clear look at the process in the form of decimal digits.

III. Modular Multiplication with Interpolation-based Multiplication

A. Modular Multiplication for Special Moduli

In cryptography, it is quite often that the modulus is chosen to be in a special form, for example, the NIST primes [17]. We choose the Curve P-521 as an example of such moduli. In Curve P-521, the modulus p is set to be $2^{521} - 1$. Then every product s of two integers less than p is less than p^2 and can be written

$$s = s_1 2^{521} + s_0, \tag{2}$$

where $0 \le s_0, s_1 < 2^{521}$. Then

$$s \mod p = (s_0 + s_1) \mod p. \tag{3}$$

We are going to show in the next section how the modular reduction can be embedded in the steps of interpolation-based multiplication in order to reduce the intermediate computations.

B. Montgomery Modular Multiplication

We present here the Montgomery modular multiplication with no deeper discussions of it, where the modulus is N, R and N' are selected parameters. [18]

Data: N, R, N'
$R \ge 4N$
$gcd(R, N) = 1$
$NN' \equiv -1 \mod R$
Input: $0 \le x, y < 2N$
Output: $z = xyR^{-1} \mod N$
$0 \le z < 2N$
$T = x \times y$
$s = (T \mod R) \times N'$
$t = (s \mod R) \times N$
$z = (t + T)/R$

Algorithm 1: Montgomery Modular Multiplication

Algorithm 1 precomputes N' with the parameter R chosen with respect to N. Usually R is set to be a power of 2 in order to simplify the operations $\mod R$ and $/R$ in Algorithm 1.

IV. Our Proposal

In cryptography, the result of modular multiplication is often the input of modular multiplication in the next round. Thus, we illustrate our example of

$$(12345678 \times 21436587 \mod 10^8 - 1) \times 13572468 \mod 10^8 - 1,$$

here using similar modulus as Curve P-521, that is $p = 10^8 - 1$ due to the fact that $12345678, 21436587, 13572468$ are both of 8 decimal digits. In the first round of modular multiplication, as we have obtained the product 264649200520986, then

$$264649200520986 \equiv (00520986 + 2646492) \equiv 3167478 \mod 10^8 - 1. \tag{4}$$

Afterwards, we perform $3167478 \times 13572468 \mod 10^8 - 1$. Similarly to the example we have performed before, let $a = 12345678, b = 21436587, s = 3167478, t = 13572468$. Then

$$\begin{aligned}
&S = \\
&\begin{pmatrix} 78 & 74 & 16 & 3 \end{pmatrix}^T \\
&T = \\
&\begin{pmatrix} 68 & 24 & 57 & 13 \end{pmatrix}^T \\
&\hat{S} = E \cdot S = \\
&\begin{pmatrix} 78 & 74 & 16 & 3 & 152 & 94 & 77 & 19 & 171 \end{pmatrix}^T \\
&\hat{T} = E \cdot T = \\
&\begin{pmatrix} 68 & 24 & 57 & 13 & 92 & 125 & 37 & 70 & 162 \end{pmatrix}^T \\
&\hat{U} = (E \cdot S) \otimes (E \cdot T) = \\
&\begin{pmatrix} 5304 & 1776 & 912 & 39 & 13984 & 11750 & 2849 & 1330 & 27702 \end{pmatrix}^T \\
&U = I \cdot ((E \cdot S) \otimes (E \cdot T)) = \\
&\begin{pmatrix} 5304 & 6904 & 7310 & 5820 & 1946 & 379 & 39 \end{pmatrix}^T \\
&u = P_7^T \cdot (I \cdot ((E \cdot S) \otimes (E \cdot T))) = \\
&42990493795704 \\
&u = s \cdot t = 3167478 \times 13572468 = \\
&42990493795704 \\
&v = u \mod p = 42990493795704 \equiv \\
&(429904 + 93795704) \equiv 94225608 \mod 10^8 - 1
\end{aligned}$$

We summarize the data flow as follows

$$\begin{aligned}
&a(b) \to A(B) \xrightarrow{E} \hat{A}(\hat{B}) \xrightarrow{\otimes} \hat{C} \\
&\xrightarrow{I} C \xrightarrow{P_7^T} c \xrightarrow{\mod p} s \to S \xrightarrow{E} \hat{S} \\
&\xrightarrow{\otimes} \hat{U} \xrightarrow{I} U \xrightarrow{P_7^T} u \xrightarrow{\mod p} v.
\end{aligned}$$

As can be directly seen from the above flow that the steps between two integer multiplication ($\xrightarrow{\otimes}$) steps are $\xrightarrow{I} C \xrightarrow{P_7^T} c \xrightarrow{\mod p} s \to S \xrightarrow{E} \hat{S}$, totalling 5 steps. Moreover, the data changes from vectors to integers, and then integers to vectors. To have a clear impression of the changes, we list the vectors and integers as below in Table II.

TABLE II
Data Flow Between Integer Multiplication Operations

Operation	Data	Description
$\xrightarrow{\otimes}$	$\hat{C}$	9-dim vector with double-word entries
$\xrightarrow{I}$	C	7-dim vector with double-word entries
$\xrightarrow{P^T}$	c	integer of 8 words
$\xrightarrow{\mod p}$	s	integer of 4 words
$\to$	S	4-dim vector with single-word entries
$\xrightarrow{E}$	$\hat{S}$	9-dim vector with single-word entries

Note: The double-word integers are no more than β^2 multiplied by a small constant. The single-word integers are no more than β multiplied by a small constant.

A. Reducing The Steps Between Integer Multiplication Operations for 4-Term Karatsuba

We present our method here directly, and show the example of such method afterwards. For the double-word vector $\hat{C}$, let $\hat{C}_L$ denote the vector of lower single words of $\hat{C}$, and $\hat{C}_H$ denote the vector of higher single words of $\hat{C}$, where the following formula holds,

$$\hat{C} = \hat{C}_L + \hat{C}_H\beta. \tag{5}$$

Then one can divide $\hat{C}$ into two vectors of single-word entries. Moreover, let M_L denote the matrix representing the operation $\bmod p$ for C_L, where the way to obtain the matrix M_L from the modulus p are to be discussed in the examples. Similarly, let M_H denote the matrix representing the operation $\bmod p$ for C_H. Furthermore, M_L, M_H are matrices of size 4×7.

The single-word vector $E \cdot S$ can then be computed as

$$\hat{S}' = (E \cdot M_L \cdot I)\hat{C}_L + (E \cdot M_H \cdot I)\hat{C}_H. \tag{6}$$

Also, if we denote $L = E \cdot M_L \cdot I$ and $H = E \cdot M_H \cdot I$, then the above formula can even be simpler

$$\hat{S}' = E \cdot S' = L \cdot \hat{C}_L + H \cdot \hat{C}_H \tag{7}$$

Therefore, the data flow between integer multiplication operations now becomes

$$\hat{C} \to \hat{C}_L(\hat{C}_H) \stackrel{L,H}{\to} \hat{S}',$$

where the step from $\hat{C}$ to $\hat{C}_L(\hat{C}_H)$ is just splitting words. Obviously, our method would reduce the 5 steps containing summation, modular reduction, integer splitting and vectors multiplied by matrices to the single step containing only vectors multiplied by matrices. Besides, the propagation in summation and modular reduction are serial operations, while our method provides a way of parallel computing since $L, H, \hat{C}_L, \hat{C}_H$ contains no data-dependency.

The most essential parts are to describe the operation $\bmod p$ as the matrices M_L, M_H and splitting the double-word vector $\hat{C}$ into single-word vectors $\hat{C}_L, \hat{C}_H$. Before the proof and formalization of our method, we show an example containing such M_L, M_H and how to compute $\hat{S}'$ from $\hat{C}_L, \hat{C}_H$.

B. An Example of Our Method

Considering $\beta = 10^2$

$$P_7 = \left(1 \ \beta \ \beta^2 \ \beta^3 \ \beta^4 \ \beta^5 \ \beta^6 \right)^T,$$

and

$$G = \left(g_0 \ g_1 \ g_2 \ g_3 \ g_4 \ g_5 \ g_6 \right)^T,$$

then we have

$$P_7^T \cdot G = g_0 + g_1\beta + g_2\beta^2 + g_3\beta^3 + g_4\beta^4 + g_5\beta^5 + g_6\beta^6.$$

Since $p = 10^8 - 1 = \beta^4 - 1$, then

$$\begin{aligned} & P_7^T \cdot G \mod p \\ =& (g_0 + g_4) + (g_1 + g_5)\beta + (g_2 + g_6)\beta^2 + g_3\beta^3 \\ =& \left(1 \ \beta \ \beta^2 \ \beta^3 \right) \cdot \begin{pmatrix} 1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&1&0&0&0&1 \\ 0&0&0&1&0&0&0 \end{pmatrix} \cdot G \end{aligned}$$

Hence, the matrix M_L is

$$M_L = \begin{pmatrix} 1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&1&0&0&0&1 \\ 0&0&0&1&0&0&0 \end{pmatrix}. \tag{8}$$

Similarly,

$$\begin{aligned} & \beta \cdot P_7^T \cdot G \mod p \\ =& g_3 + (g_0 + g_4)\beta + (g_1 + g_5)\beta^2 + (g_2 + g_6)\beta^3 \\ =& \left(1 \ \beta \ \beta^2 \ \beta^3 \right) \cdot \begin{pmatrix} 0&0&0&1&0&0&0 \\ 1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&1&0&0&0&1 \end{pmatrix} \cdot G \end{aligned}$$

$$M_H = \begin{pmatrix} 0&0&0&1&0&0&0 \\ 1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&1&0&0&0&1 \end{pmatrix}. \tag{9}$$

Naturally, we apply $L = EM_LI$ and $H = EM_HI$, then

$$L = \begin{pmatrix} 1&-1&1&-1&0&0&1&0&0 \\ -1&-1&-1&-1&1&0&0&1&0 \\ -1&1&-1&1&0&1&0&0&0 \\ 1&1&1&1&-1&-1&-1&-1&1 \\ 0&-2&0&-2&1&0&1&1&0 \\ 0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&-1&-1&0&1 \\ 0&2&0&2&-1&0&-1&-1&1 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{10}$$

$$H = \begin{pmatrix} 1&1&1&1&-1&-1&-1&-1&1 \\ 1&-1&1&-1&0&0&1&0&0 \\ -1&-1&-1&-1&1&0&0&1&0 \\ -1&1&-1&1&0&1&0&0&0 \\ 2&0&2&0&-1&-1&0&-1&1 \\ 0&0&0&0&0&-1&-1&0&1 \\ 0&0&0&0&0&1&1&0&0 \\ -2&0&-2&0&1&1&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix} \tag{11}$$

For $\hat{C}$ in the case of $12345678 \times 21436587 \times 13572468 \mod 10^8 - 1$, we rewrite it as

$$\begin{aligned} & \hat{C} = \\ & \left(6786 \ 3640 \ 1462 \ 252 \ 20368 \ 14560 \ 5848 \ 2944 \ 38880 \right)^T \\ =& \left(86 \ 40 \ 62 \ 52 \ 68 \ 60 \ 48 \ 44 \ 80 \right)^T \\ +& \left(67 \ 36 \ 14 \ 2 \ 203 \ 145 \ 58 \ 29 \ 388 \right)^T \times 100. \end{aligned}$$

Then we have

$$\begin{aligned} \hat{S}' &= L \cdot C_L + H \cdot C_H \\ &= \left(176 \ -27 \ 117 \ 202 \ 149 \ 293 \ 175 \ 319 \ 468 \right)^T \\ &= E \cdot \begin{pmatrix} 176 \\ -27 \\ 117 \\ 202 \end{pmatrix} \\ &= E \cdot S' \end{aligned}$$

Thus, the corresponding s' for $\hat{S}'$ is

$$\begin{aligned} s' &= \left(1 \ 100 \ 10000 \ 1000000 \right) \cdot \begin{pmatrix} 176 \\ -27 \\ 117 \\ 202 \end{pmatrix} = 203167476 \\ &= 3167478 + 2 \times (10^8 - 1) \\ &= s + 2p \equiv s \mod p. \end{aligned}$$

All of the above discussions have shown that, other than the difference of a multiple of p, $\hat{S}$ and $\hat{S}'$, represent the evaluation vectors of the same value s. We continue the computation with $\hat{S}'$ rather than $\hat{S}$,

$$
\begin{aligned}
&\hat{S}' = E \cdot S' = \\
&\quad \begin{pmatrix} 176 & -27 & 117 & 202 & 149 & 293 & 175 & 319 & 468 \end{pmatrix}^T \\
&\hat{T} = E \cdot T = \\
&\quad \begin{pmatrix} 68 & 24 & 57 & 13 & 92 & 125 & 37 & 70 & 162 \end{pmatrix}^T \\
&\hat{U}' = (E \cdot S') \otimes (E \cdot T) = \\
&\quad \begin{pmatrix} 11968 & -648 & 6669 & 2626 & 13708 & 36625 & 6475 & 22330 & 75816 \end{pmatrix}^T \\
&U' = I \cdot ((E \cdot S') \otimes (E \cdot T)) = \\
&\quad \begin{pmatrix} 11968 & 2388 & 17340 & 17293 & 11166 & 13035 & 2626 \end{pmatrix}^T \\
&u' = P^T \cdot (I \cdot ((E \cdot S') \otimes (E \cdot T))) = \\
&\quad 2757484066650768 \\
&v' = u' \mod p = 2757484066650768 \equiv \\
&\quad (27574840 + 66650768) \equiv 94225608 \mod 10^8 - 1
\end{aligned}
$$

That is exactly what we have computed as the result of $12345678 \times 21436587 \times 13572468 \mod 10^8 - 1$. Therefore, in the sense of modular p, $\hat{S}$ and $\hat{S}'$ are equivalent and they act identically in computations modular p.

To summarize, our method provides a one-step operation from the output of integer multiplication of the first round to the input of integer multiplication of the second round, which is much shorter and provides more parallelism compared to the original method.

C. Proof of Correctness

So far we have shown all we need for proof of correctness. We define formally the matrices M_L, M_H for n-term Karatsuba multiplications with respect to modulus p here as $n \times (2n-1)$ matrices satisfying

$$
\begin{aligned}
P_{2n-1}^T \cdot G &\equiv P_n^T \cdot M_L \cdot G \mod p \\
\beta P_{2n-1}^T \cdot G &\equiv P_n^T \cdot M_H \cdot G \mod p
\end{aligned}
$$

Then

$$
\begin{aligned}
s &= c \mod p \\
&= P_{2n-1}^T \cdot I \cdot \hat{C} \mod p \\
&= P_{2n-1}^T \cdot I \cdot (\hat{C}_L + \beta \hat{C}_H) \mod p \\
&= P_{2n-1}^T \cdot I \cdot \hat{C}_L + \beta P_{2n-1}^T \cdot I \cdot \hat{C}_H \mod p \\
&\equiv P_n^T \cdot (M_L \cdot I \cdot \hat{C}_L + M_H \cdot I \cdot \hat{C}_H) \mod p
\end{aligned}
$$

Hence, in the sense of modulo p,

$$S' = M_L \cdot I \cdot \hat{C}_L + M_H \cdot I \cdot \hat{C}_H$$

is equivalent to S. Then

$$\hat{S}' = E \cdot (M_L \cdot I \cdot \hat{C}_L + M_H \cdot I \cdot \hat{C}_H) = L \cdot \hat{C}_L + H \cdot \hat{C}_H$$

is equivalent to $\hat{S}$ in the sense of modulo p, which completes the proof.

D. Generation of Matrices M_L, M_H Corresponding to NIST Primes

We discuss here the generation of matrices M_L, M_H for NIST primes due to their great applications in cryptography. Rather than directly apply the method above, we would introduce several techniques that would help deal with more common moduli.

First of all, we show how to generate corresponding matrices for the Curve P-521 as described in the above discussions and examples. In the example applying 4-Term Karatsuba, we chose $p = \beta^4 - 1$. However, in P-521, $p = \beta^4 - 1$ means $\beta^4 = 2^{521}$, which further leads to $\beta = \sqrt[4]{2}^{521}$. Obviously, when constructing the coefficient vector A, finding a_i from a,

$$a = a_0 + a_1(\sqrt[4]{2}^{521}) + a_2(\sqrt[4]{2}^{521})^2 + a_3(\sqrt[4]{2}^{521})^3.$$

We have not found an efficient way to deal with such irrational bases. Note that we can just apply 2-Term Karatsuba, namely, the original Karatsuba to fit the modulus and let $\beta = 2^{521}$, but this method would limit the application of our proposal since there are implementations requiring smaller word length of multipliers. Instead, we provide another solution which applies 4-Term Karatsuba to P-521. We shift our modulus $p = 2^{521} - 1$ to $p' = 8p = 2^{524} - 8$ and the base can be chosen as $\beta = 2^{131}$. Under this circumstance, we have

$$a \times b \mod p = (a \times b \mod p') \mod p,$$

which is ensured by the Chinese Remainder Theorem. Therefore, the modular arithmetic can be operated modular p' and applies another reduction modular p. Considering $p' = \beta^4 - 8$, the matrix $M_L (M_H)$ of 4-Term Karatsuba for p' can then be deduced as follows,

$$
\begin{aligned}
&P^T \cdot G \mod p' \\
=&\sum_{i=0}^{6} g_i \beta^i \mod \beta^4 - 8 \\
=&(g_0 + 8g_4) + (g_1 + 8g_5)\beta + (g_2 + 8g_6)\beta^2 + g_3\beta^3 \\
=&\begin{pmatrix} 1 & \beta & \beta^2 & \beta^3 \end{pmatrix} \cdot \begin{pmatrix} 1&0&0&0&8&0&0 \\ 0&1&0&0&0&8&0 \\ 0&0&1&0&0&0&8 \\ 0&0&0&1&0&0&0 \end{pmatrix} \cdot G \\
=&\begin{pmatrix} 1 & \beta & \beta^2 & \beta^3 \end{pmatrix} \cdot M_L \cdot G
\end{aligned}
$$

Furthermore, the matrix $L = E M_L I$ for p' is

$$
L = \begin{pmatrix}
1 & -8 & 8 & -8 & 0 & 0 & 8 & 0 & 0 \\
-1 & -1 & -8 & -8 & 1 & 0 & 0 & 8 & 0 \\
-1 & 1 & -1 & 8 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 \\
0 & -9 & 0 & -16 & 1 & 0 & 8 & 8 & 0 \\
0 & -7 & 7 & 0 & 0 & 1 & 8 & 0 & 0 \\
0 & 0 & -7 & -7 & 0 & -1 & -1 & 7 & 1 \\
0 & 2 & 0 & 9 & -1 & 0 & -1 & -1 & 1 \\
0 & -7 & 0 & -7 & 0 & 0 & 7 & 7 & 1
\end{pmatrix}
$$

Hence, whether or not the bit-length of the modulus is divisible by the number of terms used in Karatsuba multiplication, we can always perform the proposed method properly via a binary shift. If we want to perform 6-Term Karatsuba multiplication for P-521, then we can choose $\beta = \lceil \frac{521}{6} \rceil = 87$ and $p' = 2p = 2^{522} - 2 = \beta^6 - 2$.

For P-192, $p = 2^{192} - 2^{64} - 1 = \beta^3 - \beta - 1$ for $\beta = 2^{64}$. Therefore, when 3-term Karatsuba multiplication is applied, M_L, M_H can also be quickly obtained via the method presented as examples,

$$
\begin{aligned}
&P_5^T \cdot G \mod p \\
=&(g_0 + g_3) + (g_1 + g_3 + g_4)\beta + (g_2 + g_4)\beta^2 \\
=&\begin{pmatrix} 1 & \beta & \beta^2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot G \\
&\beta \cdot P_5^T \cdot G \mod p \\
=&(g_2 + g_4) + (g_0 + g_2 + g_3 + g_4)\beta + (g_1 + g_3 + g_4)\beta^2 \\
=&\begin{pmatrix} 1 & \beta & \beta^2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \cdot G
\end{aligned}
$$

where M_L, M_H are uniquely determined when P^T and p are chosen.

For P-224, $p = 2^{224} - 2^{96} + 1 = \beta^7 - \beta^3 + 1$ for $\beta = 2^{32}$. Hence, 7-term Karatsuba multiplication can be applied.

For P-256, $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 = \beta^8 - \beta^7 + \beta^6 + \beta^3 - 1$ for $\beta = 2^{32}$, we apply 8-term Karatsuba multiplications.

For P-384, $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 = \beta^{12} - \beta^4 - \beta^3 + \beta - 1$ for $\beta = 2^{32}$, we apply 12-term Karatsuba multiplications.

For general moduli, we can compute the modular multiplication via Montgomery modular multiplication since the modular multiplication can be computed using multiplications, additions and truncations, where truncations are binary shifts when R is a power of 2. Consider a Montgomery modular multiplication based on n-term Karatsuba multiplications, we have $R = \beta^n$, then the operation $\mod R$ can be viewed as the following formula using similar methods mentioned above discussing M_L, M_H.

$$
\begin{aligned}
&P_{2n-1}^T \cdot G \mod R \\
=&\sum_{i=0}^{2n-2} g_i \beta^i \mod \beta^n \\
=&\sum_{i=0}^{n-1} g_i \beta^i \\
=&\begin{pmatrix} 1 & \beta & \cdots & \beta^{n-1} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \cdot G
\end{aligned}
$$

Then the matrices M_L, M_H, L, H for modulus $R = \beta^n$ can then be derived. Also, for the operation $/R$, since the division by R is exact, then, only the higher n terms of G are considered, and similarly, M_L, M_H, L, H can similarly be computed.

It can be directly deduced from the case of Montgomery modular multiplication that Barrett modular multiplication can also benefit from this method due to similar operations required in both modular multiplications. [19]

E. Complexity Analysis of Our Proposal

We analyze the complexity of both original method and our proposal when conducting Karatsuba-like multiplications in this section. First of all, we present the distinct steps, and the number of unit multiplications required by a n-term Karatsuba multiplication is denoted by $M(n)$ in the following discussions:

$$
\begin{aligned}
\text{Original} \quad & \hat{C} \xrightarrow{I} C \xrightarrow{P_{2n-1}^T} c \xrightarrow{\mod p} s \rightarrow S \xrightarrow{E} \hat{S} \\
\text{Ours} \quad & \hat{C} \rightarrow \hat{C}_L(\hat{C}_H) \xrightarrow{L(H)} \hat{S}'
\end{aligned}
$$

Since the density of the matrices I, E, L, H depends highly on choices of parameters and implementations, we are going to discuss the operations in the critical paths with maximal parallelism of both methods rather than the overall computation consumption. Both methods start from $M(n)$-dim vector $\hat{C}$ of double-word integers and end with $M(n)$-dim vector $\hat{S}$ or $\hat{S}'$ of signle-word integers. Assume that a m-word integer addition takes $ADD(m)$ of time.

In original n-term Karatsuba multiplications,

- multiplying the $(2n-1) \times M(n)$ matrix I, takes up to $M(n)ADD(2)$. Obtain C;
- multiplying the $(2n-1)$-dim vector P_{2n-1}^T, takes up to $(2n-1)ADD(1)$. Obtain c;
- modulo p, takes up to 0. Obtain s;
- splitting into the n-dim vector S, takes up to 0. Obtain S;
- multiplying the $M(n) \times n$ matrix E, takes up to $nADD(1)$. Obtain $\hat{S}$.

Hence the length of the critical path for original n-term Karatsuba multiplications is $M(n)ADD(2) + (3n-1)ADD(1)$.

In our proposal,

- splitting into two n-dim vectors $\hat{C}_L, \hat{C}_H$, takes up to 0. Obtain $\hat{C}_L, \hat{C}_H$;
- multiplying the $M(n) \times M(n)$ matrix L, H, takes up to $M(n)ADD(1)$. Obtain $L\hat{C}_L, H\hat{C}_H$;
- summing up $L\hat{C}_L, H\hat{C}_H$, takes up to $ADD(1)$. Obtain $\hat{S}'$.

Hence the length of the critical path for our proposal is $(M(n)+1)ADD(1)$.

As n grows, the ratio of improvement approaches to

$$
\lim_{n\to\infty} 1 - \frac{(M(n)+1)ADD(1)}{M(n)ADD(2) + (3n-1)ADD(1)} = 1 - \frac{ADD(1)}{ADD(2)}
$$

since $M(n)$ grows at lest subquadratically for known results of Karatsuba-like multiplications. If $ADD(2) = 2ADD(1)$ as in most implementations, the ratio becomes 50%. For $ADD(2) = 2ADD(1)$, we list the ratios for $M(n)$ in [12] in Table III, where $ADD(2)$ is denoted by α and $ADD(1)$ is denoted by γ. As can be seen from the table, our method has outperformed the original method with significant improvement (approaching 50%). Moreover, for small n, which are mostly used in impelementations, the improvement becomes even greater.

V. Future Research

A. Applications in Interpolation-based Multiplications

General interpolation-based multiplications compute the product of two polynomials with the following form,

$$
C = I \cdot ((E \cdot A) \otimes (E \cdot B))
$$

TABLE III
RATIO OF CRITICAL PATHS

n	$M(n)$	T_{orig}	T_{ours}	$1-\frac{T_{orig}}{T_{ours}}\vert_{\alpha/\gamma=2}$
1	1	$\alpha+2\gamma$	2γ	50.0%
2	3	$3\alpha+5\gamma$	4γ	63.6%
3	6	$6\alpha+8\gamma$	7γ	65.0%
4	9	$9\alpha+11\gamma$	10γ	65.5%
5	13	$13\alpha+14\gamma$	14γ	65.0%
6	17	$17\alpha+17\gamma$	18γ	64.7%
7	22	$22\alpha+20\gamma$	23γ	64.0%
8	27	$27\alpha+23\gamma$	28γ	63.6%
9	34	$34\alpha+26\gamma$	35γ	62.8%
10	39	$39\alpha+29\gamma$	40γ	62.6%

, where A, B, C are coefficient vectors for multiplicand polynomials and the product polynomial. As we can summarize from the overall paper, once the multiplication holds with some parameters I, E, then, one can always obtain

$$L = E \cdot M_L \cdot I \quad H = E \cdot M_H \cdot I$$

Thus, once the product $\hat{C} = \hat{A} \otimes \hat{B}$ is obtained, the evaluation vector for $c \mod p$ is computed simply via

$$L \cdot \hat{C_L} + H \cdot \hat{C_H}$$

In the future research of this topic, we are going to optimize I, E for interpolation-based multiplications like Toom-Cook multiplication, FFT-based multiplication and many other multiplications. [5] [8] [6]

B. Karatsuba-class Arithmetic

In the above discussions, we maintain the evaluation vectors, or Karatsbua-class defined here, for polynomials during modular multiplications. Furthermore, in public-key cryptography, modular additions and modular subtractions are frequently applied. The future research would develop Karatsuba-class arithmetic to perform all possible modular arithmetic operations without converting Karatsuba-class to normal integers or polynomials.

C. Bootstrapping and Sign-detection

As can be seen from the examples described above, entries of $\hat{C}$ are often larger than a multiple β^2. For example, in consecutive modular squaring, if there is an entry of $\hat{C}$ larger than $k\beta^2$, than the largest entry of the evaluation vector for $c \mod p$ using our method would be greater than $k\beta$. Hence, for the next round, the largest entry of the evaluation vector for $c^2 \mod p$ is larger than $k^2\beta$. The entries would grow extremely fast during iterative modular multiplications. The method can be further extended to use three matrices L, M, H, such that

$$\hat{C} = \hat{C_L} + \hat{C_M}\beta + \hat{C_H}\beta^2$$

, where $\hat{C_L}, \hat{C_M}$ are both single-word vectors, and $\hat{C_H}$ is a vector of small integers with respect to β, then

$$L \cdot \hat{C_L} + M \cdot \hat{C_M} + H \cdot \hat{C_H}$$

would be a single-word integer no larger than a constant multiple during iterative modular multiplications.

Sign-detection would be further researched for hardware implementations, since the truncations of binary integers in the form of 2's-complement are much difficult in hardware..

VI. CONCLUSION

In this paper, we presented a novel method of modular multiplication based on Karatsuba-like multiplications. This method is useful for both special moduli like NIST primes and general moduli based on Montgomery modular multiplication. Using our method, the intermediate steps between the integer multiplications necessary would be simplified as a single simple step.

REFERENCES

[1] R. L. Rivest, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the Acm*, vol. 26, no. 1, pp. 96–99, 1983.
[2] T. Elgamal, "A publi—key cryptosystem and signature scheme based on discrete logarithms. ieee transactions on information theory 31(4): 469-472," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.
[3] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.
[4] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," in *Soviet physics doklady*, vol. 7, 1963, p. 595.
[5] A. L. Toom, "The complexity of a scheme of functional elements realizing the multiplication of integers," in *Soviet Mathematics Doklady*, vol. 3, no. 4, 1963, pp. 714–716.
[6] D. D. A. Schönhage and V. Strassen, "Schnelle multiplikation grosser zahlen," *Computing*, vol. 7, no. 3-4, pp. 281–292, 1971.
[7] R. Crandall and C. Pomerance, *Prime numbers: a computational perspective*. Springer Science & Business Media, 2006, vol. 182.
[8] D. E. Knuth, "Seminumerical algorithms," 2007.
[9] D. J. Bernstein, "Multidigit multiplication for mathematicians," *Advances in Applied Mathematics*, 2001.
[10] T. Granlund, "the gmp development team. the gnu multiple precision arithmetic library manual," 2014.
[11] M. G. Find and R. Peralta, "Better circuits for binary polynomial multiplication," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 624–630, 2019.
[12] P. L. Montgomery, "Five, six, and seven-term karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362–369, March 2005.
[13] H. Fan and M. A. Hasan, "Comments on "five, six, and seven-term karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 716–717, May 2007.
[14] M. Cenk, "Karatsuba-like formulae and their associated techniques," *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 259–269, 2018.
[15] I. Oseledets, "Optimal karatsuba-like formulae for certain bilinear forms in gf (2)," *Linear Algebra and its Applications*, vol. 429, no. 8-9, pp. 2052–2066, 2008.
[16] ——, "Improved n-term karatsuba-like formulas in gf (2)," *IEEE Transactions on Computers*, vol. 60, no. 8, pp. 1212–1216, 2010.
[17] F. NIST, "Fips 186-4–digital signature standard (dss)," 2013.
[18] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.
[19] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor." in *Crypto*, vol. 86. Springer, 1986, pp. 311–323.