Fractal-Structured Karatsuba`s Algorithm for Binary Field Multiplication: FK

*The authors are working at the Institute of Mathematics in The Academy of Sciences of DPR Korea.
**Address : Un Jong district Kwahakdong Number 1 Pyongyang DPR Korea

Abstract: In this paper we report a software implementation of binary field multiplication, based on a new fractal-structured algorithm, which in practice is much faster than the current methods for the multiplication.[2,6,14,15]

Our software implementation shows that the new method archives surprising speed-up. For example, our programm on a Pentium Ⅲ-533, without optimization of compiler, by C source code edited by Microsoft Visual C++ 6.0, takes only 7.04 μs per multiplication in the field GF(2^{431}). We point out that the algorithm too suits for hardware implementation.

1. Survey of results

Recently, the study on speeding up arithmetic operations of binary field $GF(2^k)$ is very actively proceeded subject to its practical importance.[1-12]

Whereas much research successes with respect to the hardware architecture for high-speed implementation of binary field multiplication(BM) have been reported, there was no any remarkable skip after Montgomery`s method[2] in the direction of its software implementation [1-12]

In this paper we report a software implementation of BM based on a new fractal-structured algorithm, which in practice is much faster than the current methods for multiplication.[2,6,14,15] For example, in $GF(2^{431})$, this algorithm is about 7 times faster than the Montgomery multiplication[2] and more than 20 times faster than the improved standard multiplication.[6]

Our algorithm is based on the Karatsuba`s recursive subdivision(KR) method[1] for polynomial multiplication(PM).

In the practical implementation of PM, although the KR method implements less number of multiplications with smaller size in every recursive process, but it needs so much overhead that it has been recognized to be rather inefficient and disregarded till now.[13]

We have much reduced the overhead by simulating the KR process by the structure of Sierpinsky triangle, a typical example of fractal, and determining the combinatorial structure with which the assistant buffers and additions needed in every recursive step are embedded into the result of the entire process.

2. Basis of the algorithm

In this paper we employ the polynomial basis to represent elements of binary finite field $GF(2^k)$.

Lets denote the definition polynomial of a binary field $GF(2^k)$ by $n(x)$. Then we can consider the process to compute the product $c(x) = a(x) \cdot b(x) \bmod n(x)$ of $a, b \in GF(2^k)$, dividing into a PM-$a(x) \cdot b(x)$ and a modulation on the result of the PM by $n(x)$. As well known, the cost for a modulation is so cheap that it is negligible in comparison with the cost of a multiplication.

Karatsuba`s method for computing the main object-PM in speeding up BM is as following. To make the description simple, henceforth we suppose that the length of the binary finite field $k = 2^n \, (n \in N)$. Since the degrees of $a(x)$ and $b(x)$ is not beyond $k-1$, we can write as

$$a(x) = a_1(x)x^{k/2} + a_2(x), \max_i \deg a_i(x) < k/2$$

$b(x) = b_1(x)x^{k/2} + b_2(x)$, $\max_i \deg b_i(x) < k/2$ and then the PM-$a(x) \cdot b(x)$ with size k is computed by

$a \cdot b = a_1 \cdot b_1 x^k + [(a_1 + a_2) \cdot (b_1 + b_2) + a_1 \cdot b_1 + a_2 \cdot b_2] x^{k/2} + a_2 \cdot b_2$

, being converted into the combination of three PM`s-$a_1 \cdot b_1$, $a_2 \cdot b_2$, $(a_1 + a_2) \cdot (b_1 + b_2)$ with size of at most $\frac{k}{2}$. In this paper we call this converting process a Karatsuba`s recursive subdivision (KR) process. By the KR process, when $a(x) = \sum_{i=0}^{k-1} a_i x^i$ and $b(x) = \sum_{i=0}^{k-1} b_i x^i$, $a(x) \cdot b(x)$ is obtained by computing 3^n products of type $\left(\sum_{u=1}^{t} a_{i_u} \right) \cdot \left(\sum_{v=1}^{t} b_{i_v} \right)$ with unit size(Henceforth, we call these products with the least size, basic.) and combining them. What is needed is to determine or index these basic products and to find the architecture of the scheme they attend.

Linear consideration about it by a purely recursive construction seems to be rather complicate. Now, we are going to consider it on a plane, arranging the 3^i basic products in the KR process on the cells of a pre-fractal figuration gotten by the i-th action of Sierpinsky`s system of iterative functions.

To describe our method for arranging them, we first introduce some notations and definitions. For simplicity of description we denote the basic product $\left(\sum_{u=1}^{t} a_{i_u} \right) \cdot \left(\sum_{v=1}^{t} b_{i_v} \right)$ by $(i_1, \ldots, i_t)$.

When $r \in \mathbf{Z}$, we define $(i_1, \ldots, i_t) \oplus r := (i_1 + r, \ldots, i_t + r)$, $(i_1, \ldots, i_t) \oplus (j_1, \ldots, j_t) := (i_1 + j_1, \ldots, i_t + j_t)$ and denote the set consisted of all basic products in the i-th step by S_i.

Given two sets A, B composed of basic products, we define operation $\oplus$ as
$A \oplus B := \{ x \oplus y \mid x \in A,\ y \in B, \exists r \in Z, y = x \oplus r \}$ and ,when $r \in \mathbf{Z}$, $A \oplus r := \{ x \oplus r \mid x \in A \}$.

Simulating the KR process by a Sierpinsky triangle is based on a following fact.

[Lemma 1] For any $i \geq 1$

$$S_{i+1} = S_i \cup \{S_i \oplus 2^{i-1}\} \cup \{S_i \oplus \{S_i \oplus 2^{i-1}\}\},$$

where the symbol $\cup$ represents disjoint sum.

(abbreviate proof.)

□

From this lemma, we can construct three mappings:

$$\omega_1 : S_{i+1} \to S_i$$
$$\omega_2 : S_{i+1} \to S_i \oplus 2^{i-1}$$
$$\omega_3 : S_{i+1} \to S_i \oplus \{S_i \oplus 2^{i-1}\}$$

which maps all of $x, x \oplus 2^{i-1}, x \oplus (x \oplus 2^{i-1})$, where $x \in S_i$, to $x, x \oplus 2^{i-1}, x \oplus (x \oplus 2^{i-1})$, respectively.

We can consider the system of these mappings, certainly, correspondingly to the iterative function system of a Sierpinsky triangle. Namely we can completely place S_{i+1} on a $(i+1)$-size Sierpinsky triangle, by distributing the basic products corresponding to S_i in S_{i+1} on the up left triangle of size i of the $(i+1)$-size Sierpinsky triangle, basic products corresponding to $S_i \oplus 2^{i-1}$ on the up right triangle and ones corresponding to $S_i \oplus \{S_i \oplus 2^{i-1}\}$ on the down small triangle.

The just inverse of this procedure perfectly indexes all basic products.

Now, we should determine to which positions the cell corresponding to a basic product is added in the block representation of the polynomial of degree $(2^{n+1} - 2)$ obtained as the result of 2^n-size multiplication $a(x) \cdot b(x)$. This corresponds to finding the total multiplication result $a \cdot b$ from the arrangement of the basic products, considered in above discussion i. e. of the n-size pre-fractal cells in a Sierpynsky triangle. To do it, we proceed following process.

We add three $(i-1)$-size triangles in the i-size Sierpynsky triangle altogether by corresponding cells and rearrange its result (i.e. $S_i + S_i \oplus 2^{i-1} + S_i \oplus \{S_i \oplus 2^{i-1}\}$) in place of $S_i \oplus \{S_i \oplus 2^{i-1}\}$ in the i-size Sierpinsky triangle.

For any of three $(i-1)$-size pre-fractal triangles, again this procedure is iterated. … … …

Following theorem shows that our this procedure makes possible to determine the arrange situation of basic products in the product $a \cdot b$.

[Theorem 1] The result of projections of 1-size cells obtained in above procedure to the bottom side of Sierpinsky triangle is the product $a \cdot b$, where by "projection" means "Xor-addition just at the place".

(abbreviate proof.)

□

In the end, when a finite field $GF(2^k)$ is given, if we first construct a pre-Sierpynsky triangle of size n, composed of indexed basic products and then get the arrangement according to [Theorem 1] (We call this procedure a finite field $GF(2^k)$ multiplication system construction.), then we can solve the problem for computing product of any two elements a, b in the field, by computing the basic products and combining them by the constructed operation system.

This does not require any needless addition which is eliminated by Xor-addition or any delay by intermediate buffers. Having computed previously PM`s of some size v by using the table-looking-up[2], we can make the computation of basic products more simple.

3. Analysis of FK implementation

FK is designed so that for the settled problem size n select the number t of steps of KR procedure, by which we should proceed, the table size v for table-looking-up and the word size w to be optimized. In following lemma the cost for FK implementation is estimated by the number of bit-Xor operations.

[Lemma 2] [7] The cost of BM in FK is following :

$$Q(t, w, v) = \left(\frac{3}{2}\right)^t \cdot \frac{2n}{w} + \left(\frac{3}{4}\right)^t \cdot \frac{2n^2}{wv} + (n + w + 2)$$

□

The selection of v, w and t used in FK depends on following fact.

[Lemma 3] [7] The cost of PM in FK is optimized when

$$t = \min\left\{ \left[\log\left(\frac{n}{v} \cdot \frac{\log\frac{4}{3}}{\log\frac{3}{2}} \right) \right], \log\left(\frac{n}{w} \right) \right\}$$

□

In following table we show the costs of different algorithms for BM and a experimental comparison of FK to the standard multiplication and Montgomery multiplication. (In FK, when $n = 431$ the table size for reference $v = 8$, and word size $w = 32$ are selected.)

From the practical viewpoint, we have estimated the costs, supposing that all algorithms refer 8bit multiplication tables [2, 6] and proceeded the experiment in $GF(2^{431})$ on a Pentium Ⅲ-computer with Intel CPU, without optimization of compiler, by C source code edited by Microsoft Visual C++ 6.0.

Algorithm	cost	Speed of implementation in GF(2^{431}) (second/a million time)	Rate of speed-enhancement
Improved standard multiplication[6]	$34S^2 + 13S$	124	20.7
Montgomery multiplication [2]	$6S^2 + S$	42	7
FK	$\frac{1}{9}S^{\log 3} + 8S + 34$	6	1

Table 1. Comparison on the costs of FK to Montgomery and improved standard multiplication and on their implementation speeds

4. Conclusion

In this paper we have proposed a algorithm FK based on simulating the Karatsuba`s procedure for polynomial multiplication by Sierpynsky triangle, which implements the multiplication on binary fields very efficiently.

Practice has showed that FK is much faster than standard or Montgomery multiplication.

FK has been recognized to be very effective for the hardware implementation, too.

Now we are using FK as a special routine for basic operations for elliptic curve cryptosystem on binary fields.

References

[1] I. F. Blake, G. Seroussi and N. P. Smart, Elliptic Curves in Cryptography, Cambridge, U.K.: Cambridge Univ. Press , 1999.

[2] Ç. K. Koç and T. Acar, “Montgomery Multiplication in $GF(2^k)$ ”, Design, Codes and Cryptography, 14, 57-69, 1998.

[3] A. HalbutoĞullari and Ç. K. Koç. “Parallel Multiplication in $GF(2^k)$ using polynomial Residue Arithmetic”, Designs, Codes and Cryptography, 20, 155-173, 2000.

[4] M. Aydos, T. Yanik and Ç. K. Koç. “High-Speed Implementation of an ECC-based Wireless Authentication Protocol on a ARM Microprocessor”, IEE Proc. Commun., 148, 5, 273-279, 2001.

[5] R. Katti and J. Brennan, “Low Complexity Multiplication in a Finite Field Using Ring Representation”, IEEE Trans. Computers, 52, 4, 418-427,2003.

[6] Y. Han, P. C. Leong, P. C. Tan and J. Zhang. “Fast Algorithms for Elliptic Curve Cryptosystems over Binary Finite Field”, Asiacrypt`98, 75-84, 1999.

[7] S. I. Kim, G. H. Kim and C. S. Sin, “A Generalized Recursive Subdivision Method for High-Speed Implementation of Binary Field Multiplication”, Science of Information, 1, 1-3, 2005.

[8] W. Geiselmann, “A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic Circuits”, IEEE Trans. Computers, 51, 12, 1460-1461, 2002.

[9] C. H. Kim, S.Oh and J.Lim,, “A New Hardware Architecture for Operations in $GF(2^n)$”, IEEE Trans. Computers, 51, 1, 90-91, 2002.

[10] M. Elia, et al., “On the Inherent Space complexity of Fast Parallel Multipliers for $GF(2^n)$”, IEEE Trans. Computers, 51, 3, 346-351, 2002.

[11] H. Wu, ”Bit-Parallel Finite Field Multiplier and Square Using Polynomial Basis”, IEEE Trans. Computers, 51, 7, 750-758, 2002.

[12] A. Satoh and K. Takano, ”A Scalable Dual–Field Elliptic Curve cryptographic processor”, IEEE Trans. Computers, 52, 4, 449-460, 2003.

[13] R. Crandall and C. Pomerance, Prime Numbers; A Computational Perspective, Springer, p. 434, 2001.

[14] A. Reyhani-Masoleh and M. A. Hasan, “Fast Normal Basis Multiplication Using General Purpose Processors”, IEEE Trans. Computers, 52, 11, 1379-1390, 2003.

[15] D. Hankerson, J. Lopez and A. Menezes, “Software Implementation of Elliptic Curve Cryptography over Binary Fields”, CHES2000, 1-24,2000.

[16] A. Reyhani-Masoleh and M. A. Hasan, “Efficient Multiplication Beyond Optimal Normal Bases”, IEEE Trans. Computers, 52, 4, 428-439,2003.