Partially Interleaved Modular Karatsuba-Ofman Multiplication

Gokay Saldamli[†], YoJin Baek[††] and Levent Ertaul[†††],

[†]MIS Department, Bogazici University, Bebek, Istanbul, Turkey
[††]Dept. of Information Security, Woosuk University, Wanju-gun Jeonbuk 565-701, S.Korea
[†††] Department of Math & Computer Science, CSU East Bay, Hayward, CA, USA

Summary
We describe a method of performing modular multiplication that has various applications in the field of modern cryptography and coding theory. The proposed algorithm, which combines the Karatsuba-Ofman multiplier and bipartite modular reduction, presents an interleaved processing on the upper most level of Karatsuba-Ofman's recursion. The method provides an efficient and highly parallel modular arithmetic for both hardware and software realizations of public-key cryptosystems, such as today's dominating RSA and Diffie-Hellman algorithms.

Key words:
Modular multiplication; Karatsuba-Ofman multiplication; Bipartite reduction.

1. Introduction

Modular arithmetic has various applications in the field of applied sciences. This emerging trend is due to its exact integer arithmetic and the need for some advanced computations in certain mathematical structures such as finite fields, groups and rings. An apparent example is the practice of cryptography involving calculations requiring modular arithmetic.

Excluding the lighter modular addition and subtraction, research on modular arithmetic is mainly concentrated on the modular multiplication. In the finite or more specifically modular world, multiplications are carried in two steps: namely a multiplication followed by a reduction step. Since multiplication and reduction are probably the most studied two subjects in computer arithmetic, methods for carrying modular multiplication can simply be generated by matching the algorithms from each algorithm class. Obviously, this would utilize separated multiplication and reduction stages. On the other hand, a more intelligent approach would be interleaving these two operations which has a further impact on compact and scalable hardware designs.

Unfortunately, fast multipliers such as Karatsuba-Ofman (KO) [1], Furer [3] and Schonhage-Strassen [2] could not be interleaved. Hence, these area hungry asymptotically faster multipliers have to be bundled with a separate reduction process in order to realize modular multiplication (see [4] and for a recent study, we refer the reader to [5]).

Nevertheless, it is not correct to blame the fast multipliers for disabling the interleaved processing. In fact, the problem is the dependency issue of the reduction algorithms that does not permit parallel processing (i.e. parallel reduction). Bipartite modular multiplication (BMM) method introduced by Kaihara and Takagi in [6] and [7], presents a partial solution for this problem. It is based on an observation that a product could simultaneously be reduced from left and right without a dependency issue. Although, the dependency exists within each direction, BMM algorithm outlines a global method of parallel reduction.

In this study, we present a partially interleaved modular Karatsuba-Ofman multiplier based on the ideas introduced in [6] and [7]. After giving a brief description of BMM and related preliminaries, in Section 3, we outline the proposed method that merges the KO multiplier with the bipartite reduction in the upper most level of KO's recursion. The method provides an efficient and highly parallel modular arithmetic for both hardware and software realizations of today's dominating RSA and Diffie-Hellman public-key cryptosystems. From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1].

2. Modular Multiplication

Being free from the round-off errors, modular arithmetic finds various applications and studies in the field of applied sciences. While modular addition and subtraction are considered lighter, modular multiplication involving products and reductions is considered complicated. In the literature, there are various proposals and enhancements for carrying modular multiplication. A simple classification of these methods is done with respect to their reduction approach; namely, algorithms reducing from left-to-right and from right-to-left. In fact, for left-to-right approach, several proposals ([8], [9], [10], etc.) exist

Manuscript received May 5, 2015
Manuscript revised May 20, 2015

whereas Montgomery multiplication [11] is the only example of right-to-left reduction.

2.1 Left-to-right Modular Multiplication

Algorithms including the standard division can be put into this category. Assume that a, b and n are positive numbers; division algorithm states that there exist positive integers q and t (namely quotient and remainder) such that $ab = qn + t$ for $0 \leq t < n$.

Let a, b and n are represented with $k=2h$ bits for some positive integer h, the quotient q could also be considered as a k-bit number that will be written as $q= q_1 2^h + q_0$ where q_0 and q_1 are the least and most significant h bits respectively. If the a and b are partitioned into their least and most significant h bits, we would get the illustration seen in Fig. 1. Notice that either the result of a full multiplication or some intermediate partial sum can be reduced from left-to-right.

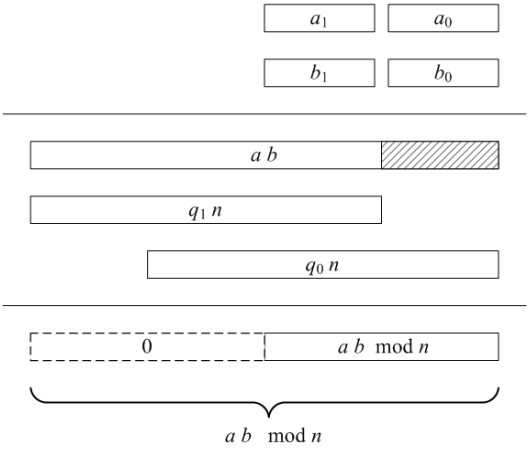

Fig. 1 Left-to-right modular multiplication.

Hardware realizations of left-to-right multiplication favorably process bit-wisely. In other words bitwise multiplication follows bitwise reduction. In general, the reduction step could involve several subtractions. However, employing an estimation logic using the few most significant bits of the partial remainder and modulus reduces this overhead to a single subtraction ([12],[13]).

2.2 Right-to-left Modular Multiplication

Montgomery approach is the only method implementing a right to left reduction (see Fig. 2). In its reduction steps, firstly, a multiple of the modulus is determined by the least significant digit of the partial sum. This multiple is then added to the partial sum in order to annihilate the least significant digit after which a trivial right shift (i.e. a reduction) is applicable.

We formally outline the Montgomery's approach as follows:

***Definition* 1:** Let n be a positive integer, and R be an integer such that $R > n$, $\gcd(n,R) = 1$. We define the n-residue of an integer a $(0 \leq a < n \cdot R)$ with respect to R as

$$\bar{a} = aR \mod n$$

Moreover we define the Montgomery reduction of a modulo n with respect to R as

$$aR^{-1} \mod n$$

Observe that Montgomery reduction and n-residues are inverses of each other. Furthermore, multiplication of two residues followed by a reduction also gives an n-residue. The following proposition describes a method of computing the Montgomery reduction carrying only trivial divisions.

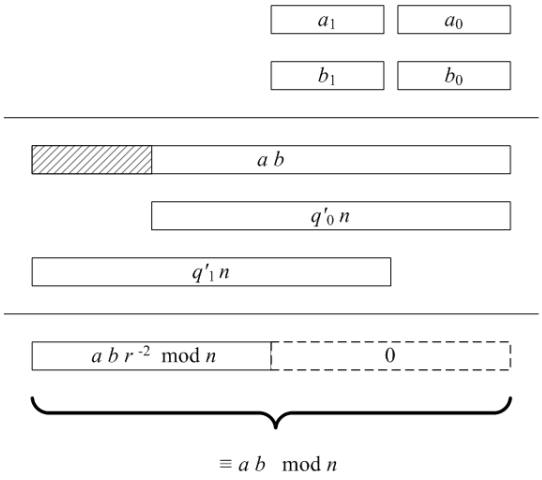

Fig. 2 Montgomery multiplication with $r^2 := R$.

***Proposition* 1**: Let a be an integer with $0 \leq a < n \cdot R$ and let $n' = -n^{-1} \mod R$. If $u = a\,n' \mod R$ then $(a + u\,n) / R$ is an integer and Montgomery reduction of a modulo n with respect to R is equivalent to

$$(a + u\,n) / R \text{ modulo } n.$$

***Proof* :** Let a and n' be as above and let $u = a\,n' \mod R$ then there exist an integers k and l such that $n'\ n = -1 + kR$ and $u = an' + lR$. This implies

$$\begin{aligned}(a + un)/R &= (a + (an' + lR)m)/R \\ &= (a + a(-1 + lR) + krn)/R \\ &= al + kn\end{aligned}$$

The second one is easier; $a \equiv (a + u\ n) \mod n \Rightarrow (a+un) / R \equiv aR^{-1} \mod n$.

The selection of R is very important in computations of n-residues and reductions. Assuming that n is an odd number, a suitable choice of R would be $R = 2^d$ for $d > \lceil \log_2(n) \rceil$. With such a choice $aR^{-1} \mod n$ can be computed with two multiplications $u = a\ n'$ and un plus some shift of $a + n$ for divisions by R.

***Remark* 1:** In order to have consistent symbols in the sections, $R = r^2$ is taken throughout the text for some positive integer r.

2.3 Bipartite Modular Multiplication

It is known that division has an intense sequential nature. This nature in the context of modular reduction comes from the dependency in determining which multiple of modulus would be added to the partial sum. Since this decision is dependent on the previously taken modulus multiples, unlike multiplication, we do not have parallel or lower complexity algorithms for reduction.

BMM method introduced by Kaihara and Takagi in [6] and [7], presents a partial solution for this problem based on an observation that a product could simultaneously be reduced from left and right without a dependency issue. Although the dependency exists within each direction, BMM algorithm outlines a global method of parallel reduction.

Assume that $r = 2^h$ and the modulus and the two respective quotients of left-to-right and right-to-left reductions are as follows;

$n := 2^h n_1 + n_0 = rn_1 + n_0,$	(1)
$q := 2^h q_1 + q_0 = rq_1 + q_0,$	(2)
$q' := 2^h q'_1 + q'_0 = rq'_1 + q'_0$	(3)

The mentioned dependency is illustrated in Figs. 1 and 2. Literally, $q_0 n$ or $q'_1 n$ are dependent to the previous reductions $q_1 n$ or $q'_0 n$ respectively. Moreover, notice from the figures that during left-to-right reduction, the shaded least significant bits of ab are not manipulated when $q_1 n$ is added to the partial sum. Similarly, the shaded part is untouched with addition of $q'_0 n$ in the right-to-left reduction. Since these two methods do not have any dependency during the first half of their reduction steps, they could be combined as seen in Fig. 3. In fact, the figure gives a sketch of the bipartite reduction.

The strength of BMM is clear; since the reduction is split into two parts, it can be handled separately in parallel. Therefore, theoretically, BMM should shrink the reduction time by half. In [7], the authors report the performance figures of BMM realizations. According to their simulations, BMM perform about 1.8 and 1.3 times better than the standard and Montgomery multipliers respectively. Obviously, these outstanding performance figures come with an area cost as reduction goes in parallel. Again in [7], the authors report that BMM needs about 1.5 and 1.7 times more area compared to the standard and Montgomery implementations respectively.

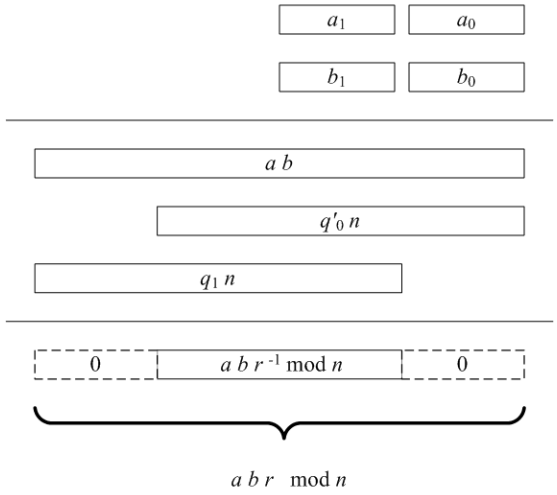

Fig.3 Bipartite reduction combining left-to-right and right-to-left reductions.

Although it is reported in [14] that further speed up on BMM is possible, this is not in the scope of our discussion. In the next section, our focus is to embed the bipartite reduction into the Karatsuba-Ofman machinery and discuss the possibility of yielding an interleaved multiplier. In fact, we demonstrate such a multiplier that operates in the upper most layer of the KO's recursion.

2. Bipartite KO Multiplier

The choice of a multiplier is subject to the application, mostly determined by the operand's bit length. For small size operands, standard multiplication is mostly used. KO multiplier is the choice for the inputs having up to a couple of thousand bits whereas big and extremely big numbers are multiplied by so called FFT techniques ([15], [16]).

On the other hand, when it comes to modular multiplication, asymptotically fast multipliers can only be utilized in a separated multiply and reduce fashion. In

reality, such a usage is not efficient as it increases the area and the total complexity. Instead, a common practice to realize compact and scalable designs is to interleave these two operations. The methods of Montgomery and Blakley are the examples of such multipliers; interleaving the standard multiplication with their specific reductions. In fact, these multipliers are the most popular ones in practice, despite the existence of the faster multipliers.

3.1 Bipartite Modular Multiplication

KO algorithm introduced in 1962 presents a recursive method which requires asymptotically fewer bit operations than the standard multiplication. In-depth details of the algorithm can be found in their original paper [1] and in Knuth[17]. For a brief explanation, firstly, we decompose a and b into two equal-size parts:

$a := 2^h a_1 + a_0,$	
$b := 2^h b_1 + b_0,$	

i.e., a_1 and a_0 represent the respective most and least significant h bits of a, assuming k is even and $2h = k$. A natural way of breaking the multiplication of a and b into multiplications of the parts a_0, a_1, b_0, and b_1 follows:

$t := ab,$	
$:= (2^h a_1 + a_0)(2^h b_1 + b_0),$	
$:= 2^{2h}(a_1 b_1) + 2^h (a_1 b_0 + a_0 b_1)\, a_0 b_0$	
$:= 2^{2h} t_2 + 2^h t_1 + t_0$	

This formulation yields the standard recursive multiplication algorithm requiring $O(k^2)$ bit operations to multiply two k-bit numbers. The KO algorithm is based on the following observation that three half-size multiplications suffice to achieve the same purpose as seen in Fig. 4:

$t_0 := a_0 b_0,$	
$t_2 := a_1 b_1,$	
$t_1 := (a_1 + a_0)(b_1 + b_0) - t_0 - t_2 = a_1 b_0 + a_0 b_1,$	

This yields to the KO recursive multiplication algorithm which requires $O(k^{1.58})$ bit operations in order to multiply two k-bit numbers. Thus, it is asymptotically faster than the standard (recursive as well as non-recursive) algorithm requiring $O(k^2)$ bit operations. Due to the recursive nature of the algorithm, there is some overhead involved. For this reason, KO algorithm starts paying off as k gets larger

Note that, one has the option of stopping at any point during the recursion. For example, we may apply one level of recursion and then compute the required three multiplications using the standard non-recursive multiplication algorithm.

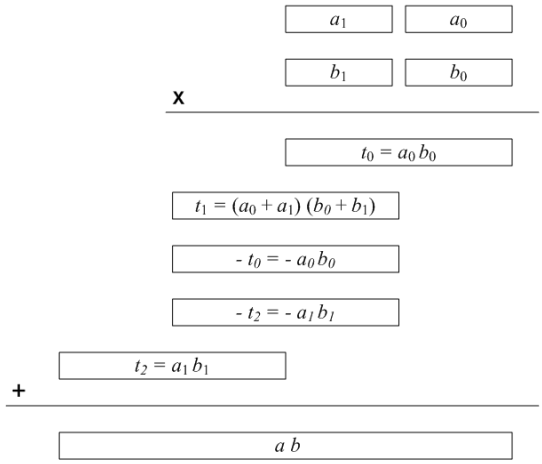

Fig.4 Karatsuba-Ofman multiplication

3.2 Interleaving modular KO multiplier

In the absence of data dependency, multiplication enjoys the bit level parallelism. Fast multipliers including KO multiplier utilize this parallelism in order to beat the quadratic complexity. As mentioned earlier, general reduction routine is not lucky in terms of data dependency. As a result, it can not be parallelized and interleaved with fast multipliers. However, to an extend, bipartite reduction could be interleaved with KO multiplier. To be specific, BMM can be interleaved with KO on the uppermost layer of KO recursion. Let q, n and q' be as in (1), (2) and (3) respectively. We start by defining the following partial products;

$t_0' r = a_0 b_0 + q_0 m_0 = t_0 + q_0' m_0$	
$t_2' = a_2 b_2 + q_1 m_1 = t_2 + q_1 m_1$	

Using these, t'_1 can be calculated as follows;

$$
\begin{aligned}
t_1' &= t_1 + t_0' + t_2' r + q_0' m_1 + q_1 m_0 \\
&= (a_0 + a_1)(b_0 + b_1) - t_0 - t_2 + t_0' r + t_2' + q_0' m_1 + q_1 m_0 \\
&= (a_0 + a_1)(b_0 + b_1) - (t_0' r - q_0' m_0) - (t_2' - q_1 m_1) \\
&\quad + t_0' r + t_2' + q_0' m_1 + q_1 m_0 \\
&= (a_0 + a_1)(b_0 + b_1) + (q_0 + q_1)(m_0 + m_1) \\
&\quad + t_0' - t_0' r + t_2' - t_2' r
\end{aligned}
$$

In fact, t'_1 gives the desired modular reduction, $abr^{-1} \bmod n$. In Fig. 5, we further illustrate how the partial products could be added to get the modular multiplication.

Proposed method's efficiency could easily be seen by counting the number of half size multiplications as tabulated in Table I. Notice from Figs. 1 and 2 that,

neglecting the expenses of lighter adds and the decision logic, a standard non-interleaved multiplier needs 8 half-size multiplications where 4 of them is for the ab generation and the other 4 is for *qn* calculation used in reduction. In case of interleaved multipliers, we still need 8 half-size multiplications but this time they are paired where a pair consists of one multiplication for reduction and another one for multiplication. Therefore, the latency drops to delay of 4 paired half-size multipliers. In fact, because of this paired organization, interleaved multipliers have compact and scalable designs.

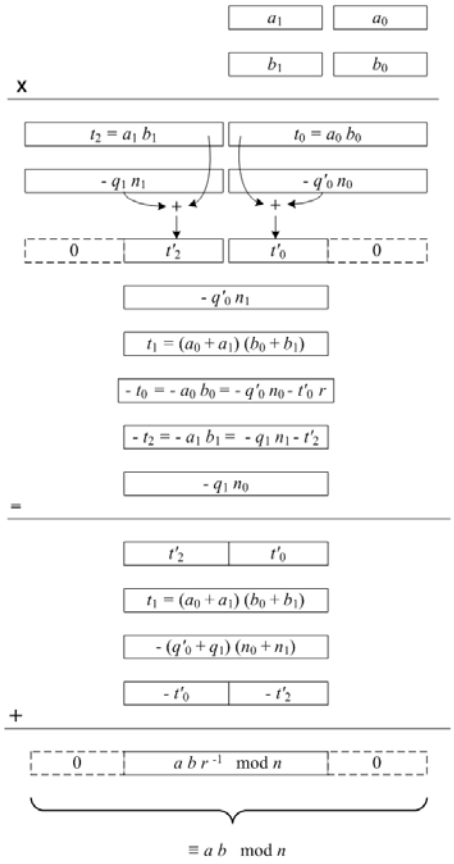

Fig.5 Partially interleaved modular KO multiplier.

On the other hand, if KO method requiring 3 half-size multiplications is used for multiplication, the number of halfsize multiplications drops to 7 for both modular multipliers bundling KO with Montgomery or BMM reductions. However, latencies differ in the existence of parallelism. First of all, since t_0, t_1 and t_2 can all be computed in parallel, the latency for KO can be taken as the latency of a single half-size multiplication. As the standard Montgomery needs 4 sequential half-size multiplications whereas BMM parallelize these; the latency becomes approximately 5 for and 3 for if the additions and subtractions are neglected.

Lastly, if the proposed method is considered, all we need to have are the following calculations:

- two paired multiplications giving t'_0, t'_2, q'_0 and also q_1.
- integer multiplications $(a_0 + a_1)(b_0 + b_1)$ and $(q'_0 + q_1)(n_0 + n_1)$

Observe that, only 6 half-size multiplications are needed for $t'_1 = abr^{-1} \mod n$ (i.e the modular multiplication) calculation.

Table 1: A simple comparison for various modular multiplications.

Algorithm	# of halfsize Multiplications (mult. + red.)	latency in halfsize multiplications
Standard	(4 + 4) = 8	8
Interleaved	(4 + 4) = 8	4*
KO + Montgomery	(3 + 4) = 7	(1 + 4) = 5
grade school +BMM	(4 + 4) = 8	(5 + 2) = 7
KO +BMM	(3 + 4) = 7	(1 + 2) = 3
proposed	(3 + 3) = 6	(1* + 1) ~ 2
Standard	(4 + 4) = 8	8
Interleaved	(4 + 4) = 8	4*
KO + Montgomery	(3 + 4) = 7	(1 + 4) = 5

* – paired

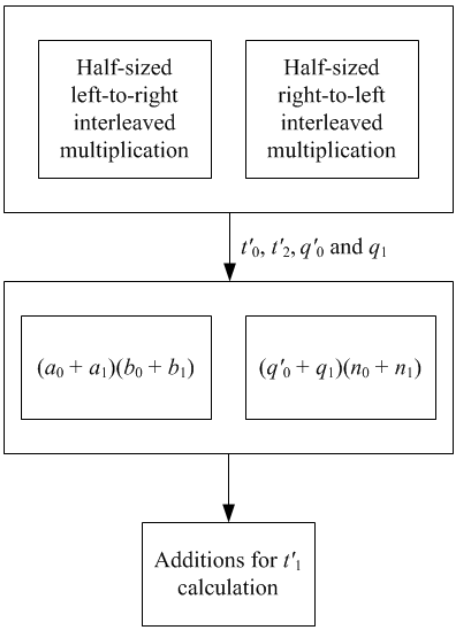

Fig.6 Block diagram of the proposed method.

As seen in the block diagram given in Fig. 6, the method can be realized in a highly parallel fashion having a delay of 1 paired and 1 half-size multiplication latency. The top block which consists of two paired multiplications (in case, able to work in parallel) implementing the idea of BMM but unlike BMM these multiplications are half-sized. Meanwhile, the block underneath implements two integer multiplications that can run in parallel. Moreover, since they are not modular, further KO recursion could be employed.

4. Conclusion

We describe a method of performing modular multiplication that combines the ideas of the bipartite modular reduction with the Karatsuba-Ofman multiplier. The method presents an interleaved processing on the upper most level of KO's recursion. However, it is not recursively applicable to the whole KO algorithm because of the sequential nature of reduction in the leaves.

We reported that with the new method, 6 half-size multiplications suffice to compute a modular multiplication of two numbers, where 8 of these are needed in case of a standard interleaved multiplier. Moreover, two of these half-sized multiplications are integer multiplications where further KO recursion could be employed. Therefore, we conclude that the presented method gives parallel architectures for hardware and software realizations of public-key cryptosystems involving modular arithmetic.

Acknowledgments

The author would like to thank Y. J. Baek from Samsung Electronics, Giheung, S. Korea for his valuable comments and discussions. G. Saldamli is partially funded by TUBITAK research project 109E180.

References

[1] A. Karatsuba, Y. Ofman, Multiplication of multidigit numbers by automata, Soviet Physics-Doklady 7 (1963) 595–596.
[2] A. Schonhage, V. Strassen, Schnelle multiplikation grafer zahlen, Computing 7 (1971) 281–292.
[3] M. Furer, Faster integer multiplication, in: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, San Diego, California, USA, 2007.
[4] C. K. Koc¸, High-Speed RSA Implementation, Tech. Rep. TR 201, RSA Laboratories, 73 pages (November 1994).
[5] G. G.W. Hasenplaugh, V. Gopal, Fast modular reduction, in: Proceedings of the 18th IEEE Symposium on Computer Arithmetic 2007 (ARITH'07), 2007, pp. 225–229.
[6] M. E. Kaihara, N. Takagi, Bipartite modular multiplication, in: Proceedings of 7th InternationalWorkshop on Cryptographic Hardware and Embedded Systems (CHES'05), 2005, pp. 201–210.
[7] M. Kaihara, N. Takagi, Bipartite modular multiplication method, IEEE Transactions on Computers 57 (2) (2008) 157–164.
[8] P. Barrett, Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor, in: Advances in Cryptology CRYPTO'86, 1987, pp. 311–323.
[9] G. R. Blakley, A computer algorithm for the product ab modulo m, IEEE Transactions on Computers 32 (5) (1983) 497–500.
[10] H. Sedlak, The RSA cryptography processor, in: Advances in Cryptology, EUROCRYPT'87, 1987, pp. 95–105.
[11] P. L. Montgomery, Modular multiplication without trial division, Mathematics of Computation 44 (170) (1985) 519–521.
[12] K. Sloan, Comments on a computer algorithm for calculating the product ab modulo m, IEEE Transactions on Computers 34 (3) (1985) 290–292.
[13] V. Bunimov, M. Schimmler, Area and time efficient modular multiplication of large integers, 14th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'03) (2003) 400.
[14] M. Knezevic, F. Vercauteren, I. Verbauwhede, Speeding up bipartite modular multiplication, in: Proceedings of the Third international conference on Arithmetic of finite fields, WAIFI'10, Springer-Verlag, 2010, pp. 166–179.
[15] R. E. Blahut, Fast Algorithms for digital signal processing, Addison-Wesley publishing Company, 1985.
[16] H. J. Nussbaumer, Fast Fourier transform and convolution algorithms, Springer, Berlin, Germany, 1982.
[17] D. E. Knuth, The Art of Computer Programming: Seminumerical Algorithms, 3rd Edition, Addison-Wesley, New York, NY, USA, 1997.

Gokay Saldamli, He is an assistant professor in Bogazici University, Turkey

YoJin Baek He is a faculty member in Dept. of Information Security, Woosuk University, South Korea

Levent Ertaul. He is currently a full time Professor at California State University Eastbay, USA in the department of Math & Computer Science.