Fast Multiplication of Integers for Public-Key Applications

Gurgen H. Khachatrian, Melsik K. Kuregian,
Karen R. Ispiryan, and James L. Massey

Cylink Corporation, 3131 Jay Street, Santa Clara, CA 95056, USA
JamesMassey@compuserve.com

Abstract. A new method for multiplication of large integers and designed for efficient software implementation is presented and compared with the well-known "schoolbook" method that is currently used for both software and hardware implementations of public-key cryptographic techniques. The comparison for the software-efficient method is made in terms of the required number of basic operations on small integers. It is shown that a significant performance gain is achieved by the new software-efficient method for integers from 192 to 1024 bits in length, which is the range of interest for all current public-key implementations. For 1024-bit integer multiplication, the savings over the schoolbook method is conservatively estimated to be about 33%. A new method for multiplication of large integers, which is analogous to the new software-efficient method but is designed for efficient hardware implementation, is also presented and compared to the schoolbook method in terms of the number of processor clock cycles required.

1 Introduction

Multiplication of large integers plays a decisive role in the efficient implementation of all existing public-key cryptographic techniques such as the Diffie-Hellman and the elliptic-curve key-agreement protocols and the Rivest-Shamir-Adelman (RSA) cryptosystem. The standard "schoolbook" method of multiplication is today the most used method for integer multiplication in practical public-key systems, cf. pp. 630-631 in [1]. For very large integers beyond the range of practical interest in current cryptographic systems, more efficient methods of multiplication are known, cf. [2]. One of these methods that has some practical significance is that due to Karatsuba and Ofman [3], which reduces the asymptotic complexity of multiplying two N-bit integers to $0(N^{1.585})$ bit operations compared to $0(N^2)$ bit operations for the schoolbook method.

The main contribution of this paper is a new software-efficient method of multiplication that improves on the schoolbook method when used in any current public-key cryptographic application. Section 2 provides a brief description of the schoolbook and the Karatsuba-Ofman methods. In Section 3 we introduce the new software-efficient method of multiplication and compare its complexity with that of the schoolbook method. We also provide explicit performance

S. Vaudenay and A. Youssef (Eds.): SAC 2001, LNCS 2259, pp. 245–254, 2001.

figures for the new soft ware-efficient method and for the schoolbook method for integers in the range from 192 to 1024 bits in length, which is the range of interest for all current public-key techniques. In Section 4 we introduce a new hardware-efficient method of multiplication, which is analogous to the new software-efficient method, and we compare its complexity in hardware with that of the schoolbook method.

2 Schoolbook and Karatsuba-Ofman Methods

Let $\beta = 2^w$ be the radix in which integers are represented for calculation. Normally, w is the word size in bits of the processor on which the algorithm is implemented. By an *n-symbol integer*, we will mean an integer between 0 and $\beta^n - 1$ inclusive, i.e., an integer that can be written as an n-place radix-β integer. Note that a *symbol* is a w-bit integer and that an n-symbol integer is an N-bit integer where $N = nw$.

Let $A = (a_{n-1}, a_{n-2}, ..a_0)$ and $B = (b_{n-1}, b_{n-2}, ..b_0)$, where a_i and b_i are w-bit integers, be two n-symbol integers. The result of their multiplication is the $2n$-symbol integer $A \cdot B$ where

$$A \cdot B = \sum_{i=0}^{n-1} a_i \beta^i \sum_{j=0}^{n-1} b_j \beta^j$$

or, equivalently,

$$A \cdot B = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \cdot b_j \beta^{i+j}. \tag{1}$$

The *schoolbook method* of multiplication computes $A \cdot B$ essentially by carrying out the n^2 multiplications of w-bit integers in (1), one for each of the n^2 terms, and adding coefficients of like powers of β. The schoolbook method thus requires n^2 multiplications of w-bit integers to calculate the product of two n-symbol integers. The precise order in which the multiplications and additions are carried out will not concern us here, but this order affects the "overhea d" in implementing the schoolbook method.

For counting the number of additions required by the schoolbook method, it is convenient first to write the $2w$-bit integer $a_i \cdot b_j$ in (1) as $c_{i,j}\beta + d_{i,j}$ where $c_{i,j}$ and $d_{i,j}$ are w-bit integers. Then (1) can be written as

$$\begin{aligned} A \cdot B &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (c_{i,j}\beta + d_{i,j})\beta^{i+j} \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{i,j}\beta^{i+j+1} + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} d_{i,j}\beta^{i+j} \end{aligned} \tag{2}$$

We note that there are only $2n$ distinct powers of β among the $2n^2$ terms in (2). Because each addition of coefficients of some power of β reduces the number of

terms by one, it follows that exactly $2n^2 - 2n = 2n(n-1)$ additions of w-bit integers are required to add the coefficients of like powers of β in (2). Thus, the schoolbook method requires $2n(n-1)$ additions of w-bit integers to calculate the product of two n-symbol integers. The additions of the terms in (2) with coefficients $c_{i,j}$ are called "carry additions" because these terms originate from the "overflow" into the next higher w-bits when two w-bit integers are multiplied. Of the $2n(n-1)$ additions of w-bit integers required by the schoolbook method, exactly half are such carry additions. Finally, we note that each addition of w-bit integers can result in a bit carry to another w-bit integer, which increments this latter integer by 1. The schoolbook method requires a maximum of $2n(n-1)$ such carry-bit additions.

The Karatsuba-Ofman method [3] is a divide-and-conquer technique for computing the components of $C = A \cdot B$ based on the following observation. Suppose that A and B are n-symbol integers where $n = 2^t$. Let $A = \beta^{2^{t-1}} A_1 + A_0$ and $B = \beta^{2^{t-1}} B_1 + B_0$ where A_0, A_1, B_0 and B_1 are 2^{t-1}-symbol integers. Then $A \cdot B = C_2 \beta^{2^t} + C_1 \beta^{2^{t-1}} + C_0$, where $C_0 = A_0 \cdot B_0$, $C_2 = A_1 \cdot B_1$, and $C_1 = (A_0 + A_1) \cdot (B_0 + B_1) - C_0 - C_2$. It follows that $C = A \cdot B$ can be computed by performing three multiplications of 2^{t-1}-symbol integers together with two additions and two subtractions of such integers. This procedure is iterated conceptually t times, i.e., until the integers reach the size of one symbol (w-bits), at which point the multiplications and additions are actually performed. This algorithm requires only $3^t \approx n^{1.585}$ multiplications of w-bit integers, compared to n^2 such multiplications for the schoolbook method. Combining Karatsuba-Ofman algorithm with schoolbook multiplication may have some practical significance. However, the recursive nature of the Karatsuba-Ofman algorithm results in such a significant overhead that its direct application to integers of the size used in current public-key cryptography is not efficient, cf. pp. 630-631 in [1].

3 A Software-Efficient Multiplication Method

3.1 The Underlying Idea

Our new software-efficient multiplication method is based on the formula

$$A \cdot B = \sum_{u=1}^{n-1} \sum_{v=0}^{u-1} (a_u + a_v) \cdot (b_u + b_v) \beta^{u+v} + 2 \sum_{u=0}^{n-1} a_u \cdot b_u \beta^{2u} - \sum_{v=0}^{n-1} \beta^v \sum_{u=0}^{n-1} a_u \cdot b_u \beta^u. \quad (3)$$

We will use the same notation here as we used for the schoolbook method except that we will write the radix as $\beta = 2^W$ rather than as $\beta = 2^w$ for a reason that will become apparent in Subsection 3.2.

It is easy to check by multiplying out and combining terms that (3) gives the correct result for multiplication. We note here for future use that $n(n-1)/2$ additions of W-bit integers are required to form the coefficients $a_u + a_v$ in (3) and another $n(n-1)/2$ such additions are required to form the coefficients $b_u + b_v$. To facilitate the counting of multiplications and further additions of W-bit integers

needed to implement the multiplication formula (3), it is convenient to write

$$a_u + a_v = a_{u,v}^{\text{cb}}\beta + a_{u,v}^{\text{sum}}$$

where $a_{u,v}^{\text{cb}}$ is the carry bit and $a_{u,v}^{\text{sum}}$ is the least significant W-bits of the sum of the W-bit integers a_u and a_v. Using analogous notation for the sum of the W-bit integers b_u and b_v, we can write (3) in the manner

$$\begin{aligned} A \cdot B = &\sum_{u=1}^{n-1}\sum_{v=0}^{u-1}(a_{u,v}^{\text{cb}}\beta + a_{u,v}^{\text{sum}}) \cdot (b_{u,v}^{\text{cb}}\beta + b_{u,v}^{\text{sum}}) \cdot \beta^{u+v} \\ &+2\sum_{u=0}^{n-1} a_u \cdot b_u\beta^{2u} - \sum_{v=0}^{n-1}\beta^v \sum_{u=0}^{n-1} a_u \cdot b_u\beta^u \end{aligned}$$

or, equivalently,

$$\begin{aligned} A \cdot B = &\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} a_{u,v}^{\text{cb}} b_{u,v}^{\text{cb}}\beta^{u+v+2} \\ &+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1}(b_{u,v}^{\text{cb}} a_{u,v}^{\text{sum}} + a_{u,v}^{\text{cb}} b_{u,v}^{\text{sum}})\beta^{u+v+1} \\ &+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} a_{u,v}^{\text{sum}} \cdot b_{u,v}^{\text{sum}}\beta^{u+v} \\ &+2\sum_{u=0}^{n-1} a_u \cdot b_u\beta^{2u} \\ &-\sum_{v=0}^{n-1}\beta^v \sum_{u=0}^{n-1} a_u \cdot b_u\beta^u. \end{aligned} \tag{4}$$

The only multiplications of W-bit integers occur within the third, fourth and fifth lines in (4). Each of the $\binom{n}{2} = \frac{n(n-1)}{2}$ terms in the third line requires one such multiplication. Each of the n terms within the sum on u in the fourth line also requires one such multiplication and these are the same products as are required in the fifth line. Thus, to implement the multiplication formula (3) requires a total of $\frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}$ multiplications of W-bit integers, which we note is about half that required by the schoolbook method when we choose $W = w$ as is required for a direct comparison.

In counting additions of W-bit integers, we consider the worst case where all the carry bits $a_{u,v}^{\text{cb}}$ and $b_{u,v}^{\text{cb}}$ are equal to 1. It is again convenient to write the $2W$-bit integer $a_u \cdot b_u$ in (4) as $c_u\beta + d_u$ where c_u and d_u are W-bit integers, and to write $a_{u,v}^{\text{sum}} \cdot b_{u,v}^{\text{sum}}$ in (4) as $c_{u,v}^{\text{sum}}\beta + d_{u,v}^{\text{sum}}$ where $c_{u,v}^{\text{sum}}$ and $d_{u,v}^{\text{sum}}$ are W-bit integers. We can then rewrite (4) for this worst case as

$$A \cdot B = \sum_{u=1}^{n-1}\sum_{v=0}^{u-1}\beta^{u+v+2}$$

$$
\begin{aligned}
&+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} a_{u,v}^{\mathrm{sum}}\beta^{u+v+1} + \sum_{u=1}^{n-1}\sum_{v=0}^{u-1} b_{u,v}^{\mathrm{sum}}\beta^{u+v+1}\\
&+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} c_{u,v}^{\mathrm{sum}}\beta^{u+v+1} + \sum_{u=1}^{n-1}\sum_{v=0}^{u-1} d_{u,v}^{\mathrm{sum}}\beta^{u+v}\\
&+\sum_{u=0}^{n-1}(c_u + c_u)\beta^{2u+1} + \sum_{u=0}^{n-1}(d_u + d_u)\beta^{2u}\\
&-\sum_{u=0}^{n}\sum_{v=0}^{n-1}(c_{u-1} + d_u)\beta^{u+v}
\end{aligned}
$$

with the convention that $c_j = d_j = 0$ for $j < 0$ and for $j \geq n$. Upon setting $e_u = c_{u-1} + d_u$ and then $u = i - v$ in the last line, we can rewrite this equivalently as

$$
\begin{aligned}
A \cdot B = &\sum_{u=1}^{n-1}\sum_{v=0}^{u-1}\beta^{u+v+2}\\
&+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} a_{u,v}^{\mathrm{sum}}\beta^{u+v+1} + \sum_{u=1}^{n-1}\sum_{v=0}^{u-1} b_{u,v}^{\mathrm{sum}}\beta^{u+v+1}\\
&+\sum_{u=1}^{n-1}\sum_{v=0}^{u-1} c_{u,v}^{\mathrm{sum}}\beta^{u+v+1} + \sum_{u=1}^{n-1}\sum_{v=0}^{u-1} d_{u,v}^{\mathrm{sum}}\beta^{u+v}\\
&+\sum_{u=0}^{n-1}(c_u + c_u)\beta^{2u+1} + \sum_{u=0}^{n-1}(d_u + d_u)\beta^{2u}\\
&-\sum_{i=0}^{2n-1}[\sum_{v=0}^{n-1} e_{i-v}]\beta^{i}. \qquad (5)
\end{aligned}
$$

The terms $e_i = c_{i-1} + d_i$ for $i = 0,\ 1,\ \ldots\ ,\ n$ require $n - 1$ additions of W-bit integers for their formation because the terms for $i = 0$ and $i = n$, namely d_0 and c_{n-1} respectively, require no additions. We next consider the number of additions of W-bit integers required to form the coefficients

$$s_i = \sum_{v=0}^{n-1} e_{i-v} \ \text{ for } i = 0,\ 1,\ \ldots\ ,\ 2n-1$$

that appear in the fifth line of (5). We observe that we can rewrite this sum separately over two ranges of the index as

$$s_i = \sum_{j=0}^{i} e_i = s_{i-1} + e_i \ \text{ for } i = 0,\ 1,\ \ldots\ n-1 \qquad (6)$$

and

$$s_{2n-1-i} = \sum_{j=0}^{i} e_{n-j} = s_{2n-i} + e_{n-i} \ \text{ for } i = 0,\ 1,\ \ldots\ n-1. \qquad (7)$$

where we have taken $s_0 = s_{2n} = 0$. We see that $n-2$ additions of W-bit integers are required to form the nested sums in (6) and another such $n-2$ additions are required to form the nested sums in (7). Hence a total of $3(n-1)$ additions of W-bit integers are required to form all the coefficients in the fifth line of (5).

The summation in the first line of (5) concerns only carry bits, which we will consider later. There are $n(n-1)$ terms in the second line, another $n(n-1)$ terms in the third line, $4n$ terms in the fourth line, and $2n$ terms in the fifth line—a total of $2n^2 + 4n$ terms. But there are only $2n$ distinct powers of β in (5) so that $2n^2 + 4n - 2n = 2n^2 + 2n$ additions of W-bit numbers are required to combine the like powers of β. To this, we must add the $n(n-1)$ additions of W-bit integers required to form the coefficients $a_u + a_v$ and $b_u + b_v$ in (3) as well as the $3(n-1)$ additions required to form the coefficients in the last line of (5). This gives a total of $3n^2 + 4n - 3$ additions of W-bit integers required to implement the multiplication formula (3). We note that this is greater by a factor of about $\frac{3}{2}$ than the $2n(n-1)$ additions required by the schoolbook method when we choose $W = w$ as is required for a direct comparison.

Finally, we note that the first line of (5) specifies $\frac{1}{2}n(n-1)$ additions of carry bits (in this worst case) and each of the $3n^2+4n-3$ additions of W-bit numbers can also result in a carry bit. Thus, to implement the multiplication formula (3) requires a maximum of $\frac{7}{2}n(n+1) - 3$ carry-bit additions.

3.2 Achieving Efficiency

As we have just seen, the direct implementation of formula (3) for multiplication of nW-bit integers requires only about half as many multiplications, but about 50% more additions, of W-bit integers compared to the schoolbook method when we take $W = w$. To convert the multiplication formula (3) into an efficient method for multiplication of N-bit integers on a w-bit processor, we first set $N = nsw$ and then split the problem of multiplication into (1) the problem of multiplying N-bit integers using a virtual processor with word size $W = sw$, followed by (2) the problem of implementing the necessary multiplications and additions of W-bit integers using the actual processor with word size w. We solve the first problem by implementing the multiplication formula (3), after which we solve the second problem by implementing the necessary multiplications of sw-bit integers by the schoolbook method. We now count the number of multiplications and additions of w-bit integers required by this "hybrid method" for multiplying N-bit integers.

As was shown in Subsection 3.1, the multiplication of N-bit integers, where $N = nW$, according to the multiplication formula (3) requires $\frac{n(n+1)}{2}$ multiplications of W-bit integers where $W = sw$. Each such multiplication when performed by the schoolbook method requires s^2 multiplications and $2s(s-1)$ additions of w-bit numbers, as well as $2s(s-1)$ carry-bit additions. Thus, *the multiplications performed in the first step of the hybrid method* result intal numbers of w-bit integer operations shown in the following table: The multiplication of N-bit integers, where $N = nW$, according to the multiplication formula (3) requires $3n^2 + 4n - 3$ additions of W-bit integers where $W = sw$. Each such addition is

multiplications	additions	carry-bit additions
$\frac{1}{2}s^2n(n+1)$	$s(s-1)n(n+1)$	$s(s-1)n(n+1)$

equivalent to s additions of w-bit integers and s carry-bit additions. Thus, the *the additions performed in the first step of the hybrid method* result in the total numbers of w-bit integer operations shown in the following table: Finally, the

multiplications	additions	carry-bit additions
0	$(3n^2+4n-3)s$	$(3n^2+4n-3)s$

multiplication of N-bit integers, where $N = nW$, according to the multiplication formula (3) requires in the worst case $\frac{7}{2}n(n+1)-3$ carry-bit additions for W-bit integers where $W = sw$. Each such carry-bit addition for sw-bit integers is equivalent to a single carry-bit addition for w-bit integers so that *the carry-bit additions performed in the first step of the hybrid method* result in the total numbers of w-bit integer operations shown in the following table:

multiplications	additions	carry-bit additions
0	0	$\frac{7}{2}n(n+1)-3$

Tallying the counts in the three previous tables gives the figures shown in the following table:

For ease of comparison, we include here the table of counts for schoolbook method as calculated in Section 2.

3.3 Numerical Examples

Example 1: Consider the multiplication of 1024-bit integers [where we note that 1024 is a length commonly used for current implementations of the RSA cryptosystem and of the Diffie-Hellman key agreement protocol] on a processor with word size $w = 16$ bits. As a basis for comparison, we assume that one 16-bit addition constitutes 1 unit of computation as also does one carry-bit addition, but that one 16-bit multiplication constitutes 2 units of computation.

The specifications $N = nsw = 1024$ and $w = 16$ give $ns = 64$ and hence the allowed values of (n, s) are $(1, 64)$, $(2, 32)$, $(4, 16)$, $(8, 8)$, $(16, 4)$, $(32, 8)$ and $(64, 1)$. Calculating the cost for each of these choices with the aid of the values in Table 1 shows that the choice $n = s = 8$ yields the minimum cost of 16,457 computational units for the new software-efficient method, but the choice $n = 4$ and $s = 16$ is nearly as good with a cost of 16,739 units. For the choice $n = s = 8$, the number of multiplications, additions and carry-bit additions are 2304, 5800 and 6049, respectively. By comparison, we calculate from Table 2 that the

Table 1. Total counts of w-bit integer operations for the software-efficient multiplication method for nsw-bit integers

multiplications	additions	carry-bit additions
$\frac{1}{2}s^2n(n+1)$	$s[(s+2)n^2+(s+3)n-3]$	$s[(s+2)n^2+(s+3)n-3]$ $+\frac{7}{2}n(n+1)-3$

Table 2. Total counts of w-bit integer operations for the schoolbook multiplication method for nsw-bit integers

multiplications	additions	carry-bit additions
$(sn)^2$	$2sn(sn-1)$	$2sn(sn-1)$

schoolbook method has a cost of 24,320 computational units arising from the 4096 multiplications, 8064 additions, and 8064 carry-bit additions that must be performed. In this example, the new software-efficient multiplication method uses about one-third less computation than does the schoolbook method.

Example 2: Consider the multiplication of 192-bit integers [which is one of the lengths for the Elliptic curve system recommended for the FIPS 186-2 standard] on a processor with word size $w = 8$ bits. Again we assume that one 8-bit addition or one carry-bit addition constitutes 1 unit of compution, but that one 8-bit multiplication constitutes 2 units of computation.

The specifications $N = nsw = 192$ and $w = 8$ give $ns = 24$ and hence the allowed values of (n, s) are $(1, 24)$, $(2, 12)$, $(3, 8)$, $(4, 6)$, $(6, 4)$, $(8, 3)$, $(12, 2)$ and $(24, 1)$. Calculating the cost for each of these choices with the aid of the values in Table 1 shows that the choice $n = 4$ and $s = 6$ yields the minimum cost of 2719 computational units for the new software-efficient method, but the choice $n = 3$ and $s = 8$ is virtually as good with a cost of 2727 units. For the choice $n = 4$ and $s = 6$, the number of multiplications, additions and carry-bit additions are 360, 966 and 1033, respectively. By comparison, we calculate from Table 2 that the schoolbook method has a cost of 3360 computational units arising from the 576 multiplications, 1104 additions, and 1104 carry-bit additions that must be performed. In this example, the new software-efficient multiplication method uses about 19% less computation than does the schoolbook method.

It should be pointed out that *actual performance results for the new software-efficient multiplication method may well be substantially better than predicted by our analysis*, which was made using worst-case assumptions. For instance, our 8-bit implementation of the new software-efficient multiplication method on a Pentium 2 processor for the parameters of Example 2 actually used about 40% less computation than did the schoolbook method, rather than only 19% less as our analysis had predicted.

4 A Hardware-Efficient Multiplication Method

The following formula, analogous to (3), is the basis for our new hardware-efficient method of multiplication:

$$A \cdot B = (\sum_{v=0}^{n-1} \beta^v)(\sum_{u=0}^{n-1} a_u \cdot b_u \beta^u) + \sum_{u=1}^{n-1} \sum_{v=0}^{u-1} (a_u - a_v) \cdot (b_v - b_u)\beta^{u+v}. \quad (8)$$

The complexity of implementing multiplication according to (8) is comparable to that for implementing multiplication according to (3). Which method is superior depends on the computational environment. For example, using (8) will give fewer carry-bit additions but will require sign checks. In general the use of (8) is better suited to hardware implementations and therefore we now analyze the use of (8) in a hardware implementation by estimating the number of clock cycles needed to multiply two N-bit numbers A and B. We will compare this performance to a hardware implementation of the schoolbook method using the shift-and-add technique, which requires N clock cycles when all N bits can be processed in parallel.

Let $N = nW$ and consider the multiplication formula (8) where a_i and b_i are W-bit numbers. Calculating all $n(n-1)$ required differences $(a_i - a_j)$ and $(b_i - b_j)$ in the second double summation of (8) requires $\frac{n(n-1)}{n} = n - 1$ clock cycles if the same resources as for the schoolbook method are used. Formula (8) requires $\frac{n(n+1)}{2}$ multiplications of W-bit numbers. Because n such multiplications can be performed in parallel, another $W\lceil\frac{(n+1)}{2}\rceil$ clock cycles are needed. Summing the results of these multiplications requires in the worst case an additional $\frac{n(n+1)}{2}$ clock cycles. The total number of clock cycles required for the multiplication $A \cdot B$ is thus

$$W(\lceil\frac{(n+1)}{2}\rceil) + \frac{n(n+1)}{2} + n - 1, \quad (9)$$

which is about half that required by the schoolbook method for large $N = nW$.

Example 3: Suppose that A and B are 1024 bit numbers and consider the choice $W = 128$ and $n = 8$. Multiplication according to (9) requires at most 683 clock cycles compared to 1024 clock cyles for the schoolbook method using the shift-and-add technique, a reduction of 33%.

5 Conclusion

The analyses of the new software-efficient multiplication method and of the new hardware-efficient multiplication method both show that a significant performance improvement over the schoolbook method can be obtained for all current applications in public key cryptography. Moreover, our complexity estimates for the new methods are conservative–actual gains can exceed those predicted, as was pointed out in Example 2.

References

1. A. Menezes, P. van Oorschot, S. Vanstone: Handbook of Applied Cryptography. CRC press, Boca Raton and New York (1997).
2. D. E. Knuth: The Art of Computer Programming, Vol. 2, 2nd Ed. Addison-Wesley, Reading, Mass. (1981).
3. A. Karatsuba, Yu. Ofman: Multiplication of Multidigit Numbers on Automata. Soviet Physics Doclady, **7**, (1963) 595-596.