



VLSI Research Group

University of Windsor

---

*Complexity and fast algorithms  
for multi-exponentiations*

**Final to IEEE Transactions on Computers**

V. S. Dimitrov, G. A. Jullien and W. C. Miller

---

# Complexity and fast algorithms for multi-exponentiations

V. S. Dimitrov, G. A. Jullien and W. C. Miller

VLSI Research Group, University of Windsor  
Windsor, Ontario, Canada N9B 3P4

## Abstract

*In this paper we propose new algorithms for multiple modular exponentiation operations. The major aim of these algorithms is to speed up the performance of some cryptographic protocols based on multi-exponentiation. Our new algorithms are based on binary-like complex arithmetic, introduced by Pekmestzi and generalized in this paper.*

**Keywords:** *Modular Exponentiation; Cryptography; Complex Arithmetic*

## 1 Introduction

Most number-theoretic cryptosystems [1]-[5] are based on modular exponentiation (ME) which requires a large number of processing steps. Therefore a significant problem is how to reduce the time needed to perform a modular exponentiation operation. The problem can be posed as follows: given  $A, B$  and  $p$ , compute  $C = A^B \pmod{p}$ . Theoretically, the problem is closely connected to addition chains. In general, an addition chain for a given positive integer,  $B$ , is a sequence of positive integers  $a_0=1, a_1, a_2, \dots, a_r=B$ , where  $r$  is the number of additions and for all  $i=1, 2, \dots, r$ ,  $a_i = a_j + a_k$  for some  $k \leq j < i$ . Any addition chain corresponds to some algorithm for the evaluation of  $A^B \pmod{p}$ . The shortest addition chains produce optimal algorithms for ME (in terms of the number of multiplications); however, obtaining at least one of the shortest addition chains for a given integer is an NP-complete problem [6]. A typical technique for computing an addition chain with a fairly small length, is based on the binary decomposition of  $B$  [12].

Obviously the number of modular multiplications (MM) strongly depends on the Hamming weight of  $B$ . Since the average number of ones in an  $n$ -bit number is equal to  $n/2$ , then the average number of multiplications required is  $\frac{3}{2}n$ .

There are several algorithms that reduce the number of multiplications to  $n + o(n)$ , namely, those proposed by Valski [7], Yao [8], Pippenger [9], Yacobi [10] and Kochergin [11]; the technique by Kochergin [11] requires  $n + \frac{n}{\log n} + \frac{\log n}{\log \log n} + \left( o\left( \frac{\log n}{\log \log n} \right) \right)$  MMs, and is the best of the quoted literature. Notwithstanding their optimality, these algorithms are mainly of theoretical interest, partly because of their irregular nature, and partly because some of them assume a ‘free’ complex preprocessing of  $B$  (say, factorizing  $B$  before proceeding further [12]).

There are several algorithms based on different (non-binary) representations of  $B$  [13]-[16]. The most popular are, perhaps, the algorithms using a signed-digit binary representation of  $B$ ; that is, representing  $B$  in the form:

$$B = \sum_{i=0}^{n-1} d_i 2^i \quad d_i \in \{0, 1, -1\} \quad (1)$$

This general representation is redundant but the canonic signed-digit binary representation (CSDBR) is unique, and there are several proofs [14], [17]-[19] showing that, on average,  $2/3$  of the digits are zeros. Using a CSDBR corresponds to addition-subtraction chains [20]-[21] with a rule  $a_i = a_j \pm a_k$  in place of  $a_i = a_j + a_k$ . This idea leads us to the evaluation of  $A^B$  using multiplications and divisions. For integers, division (or multiplication by the multiplicative inverse modulo  $p$ ) is a costly operation. However, in some cryptosystems,  $A$  is a constant, so  $A^{-1} \bmod p$  can be precomputed in advance. Another argument in favour of addition-subtraction chains comes from so-called elliptic curve cryptosystems [19], [21]-[23], where the division in  $Z_p$  is replaced by a subtraction, which has the same cost as addition.

A detailed survey of fast exponentiation methods has been published in [24]. Recently, several cryptographic protocols using multiple modular exponentiations (or multi-exponentiations)

have been proposed. The main operation here is the computation of  $z = \prod_{i=1}^k x_i^{e_i} \pmod{p}$ , for

$k > 1$ . Three of the currently proposed cryptographic protocols use  $k=2$  [4][5][25][37][38][39]; an example with  $k=3$  can be found in [3].

In this paper we concentrate our attention on the case  $k=2$ ; that is, given integers  $m$  and  $n$ , compute  $z = x^m y^n \pmod{p}$ , where  $p$  is a prime. In terms of addition chains, this can be associated with vector addition chains (VAC), as defined by Strauss [26].

It has been recognized [30], that the separate computation of powers:

$$(z = (x^m \pmod{p}) \cdot (y^n \pmod{p})) \pmod{p}$$

is not optimal even if one uses optimal algorithms for the computation of  $x^m \pmod{p}$  and  $y^n \pmod{p}$ . A very simple example is the following: the computation of  $z = x^2 y^2 \pmod{p}$  based on separate multiplications needs 3 MMs, whereas the same can be computed as  $z = (x \cdot y)^2 \pmod{p}$  requiring only 2 MMs.

It is worth while pointing out that the fastest algorithms for discrete logarithms over finite fields, proposed by Gordon [40] and Adleman-Demarrais [41], are based on modular multi-exponentiation with fixed bases. Any improvement in the computational time of this operation will immediately result in improving the performance of these algorithms. Some algorithms in

computer algebra, coding theory and control theory also require the computation of multi-exponentiations [27][28][29].

This paper is organized as follows. In Section 2 we define the problem of finding the vector addition chain, in the language of Gaussian integers, using an efficient representation of complex numbers [31]. Section 3 introduces the use of signed-digit complex arithmetic; this number representation has some features that are significantly different from CSDBR. Section 4 demonstrates the computation of multi-exponentiations via signed-digit complex arithmetic. Section 5 concludes the paper.

## 2 An Algorithm Based on Gaussian integers

Gaussian integers are complex numbers of the form  $a + bi$ , where  $a$  and  $b$  are integers. They have been intensely studied in number theory and find many applications in digital signal processing algorithms.

The problem of computing  $z = x^m y^n \pmod{p}$  is equivalent to finding some vector addition chain for the vector  $(m, n)$ . In terms of Gaussian integers it can be stated as follows: given a Gaussian integer  $\alpha = m + ni$ , find a set of Gaussian integers  $\alpha_0 = (1, 0)$ ,  $\alpha_1 = (0, 1)$ ,  $\alpha_2 = (1, 1), \dots, \alpha_r = (m, n)$ , such that for every  $l$  there exist integers  $k$  and  $j$ ,  $l > k \geq j$  such that  $\alpha_l = \alpha_k + \alpha_j$ . We will refer to this as the *Gaussian addition chain*. Therefore, the double-exponentiation problem consists of finding a sufficiently short Gaussian addition chain.

In 1989 Pekmestzi [31] introduced the following ‘binary-like’ representation of complex numbers:

$$z = \sum_j d_j 2^j \quad (2)$$

where  $d_j \in (0, 1, i, 1 + i)$ . This number representation can be considered as a binary representation (the base is 2) with complex digits. Similar representations have been studied by various authors [32]-[35]. The digit encoding [31] is shown in Table 1:

**Table 1. Digit encoding of complex digits**

Complex Digit	Binary Code
0	00
1	10
$i$	01
$i+1$	11

As an example of using the representation in eqn. (2), consider  $s = 4893 + 5096i$ . We have:

$$4893 = (1001100011101)_2$$

$$5096 = (1001111100000)_2$$

that is  $s = (1+i, 0, 0, 1+i, 1+i, i, i, i, 1, 1, 1, 0, 1)$ . If we use the above encoding scheme for the complex digits, we will have the following representation of  $s$ :

$$s = (11000011110101011010100010).$$

## 2.1 Algorithm I

Using Pekmestzi's arithmetic we now propose the following algorithm:

*Input:*  $m, n, x, y$  and  $p$ : positive integers,  $p$ -prime;

*Output:*  $z = x^m y^n \pmod{p}$

**Step 1:** Find the representation of  $s = m + ni$  in the form of eqn. (2). Encode the complex digit according to Table 1.  $s$  is represented as a binary string of length  $2h$ , where  $h = \max(\lfloor \log m \rfloor, \lfloor \log n \rfloor)$ . We denote this string as  $H = (g_{2h-1}g_{2h-2}\cdots g_1g_0)$ .

**Step 2:**  $z = 1; q = x * y \pmod{p}$ ;

**Step 3:**  $k = 2h - 1$ ;

**Step 4:** If  $(g_k, g_{k-1}) == (1, 0)$  then  $z = z * x \pmod{p}$   
 If  $(g_k, g_{k-1}) == (0, 1)$  then  $z = z * y \pmod{p}$   
 If  $(g_k, g_{k-1}) == (1, 1)$  then  $z = z * q \pmod{p}$   
 If  $(k > 2)$   $z = z * z \pmod{p}$ .

**Step 5:**  $k = k - 2$ .  
 If  $(k > 0)$  go to Step 4  
 Otherwise Output  $z$ .

This algorithm produces a corresponding Gaussian addition chain for a given Gaussian integer. Using the above example (Gaussian integer  $s=4893+5096i$ ), we have the following addition chain requiring 22 modular multiplications.

$$(0,1),(1,0),(1,1),(2,2),(4,4),(8,8),(9,9),(18,18),(19,19),(38,38),(38,39),(76,78), \\ (76,79),(152,158),(152,159),(304,318),(305,318),(610,636),(611,637), \\ (1222,1274),(1223,1274),(2446,2548),(4892,5096),(4893,5096).$$

This is considerably faster than the 36 modular multiplications required if one uses separate modular exponentiations for computing  $x^{4893} \pmod{p}$ ,  $y^{5096} \pmod{p}$  and their product modulo  $p$ .

The complexity analysis of this algorithm is straightforward: the worst case appears when all of the complex digits of  $s = m + ni$  are nonzero and in this case we have  $2h = 2\lfloor \log(\max(m, n)) \rfloor$  modular multiplications. On average  $1/4$  of the complex digits of  $s$  are expected to be zero, so the average number of MMs is  $\frac{7}{4}h = \frac{7}{4}\lfloor \log(\max(m, n)) \rfloor$ . It is clear that multi-exponentiation, based on separate exponentiations using the binary method, needs on average  $3\log(\max(m, n))$  exponentiations. Even the use of optimal algorithms for separate exponentiations will require  $2\log(\max(m, n))$  MMs, to perform multi-exponentiation, so the method presented here is clearly superior.

Algorithm 1 is of the same complexity as the algorithm for multi-exponentiations demonstrated in [42], pp.618. In the next section we will generalize this complex arithmetic to obtain an algorithm with lower complexity.

### 3 A signed-digit complex arithmetic

The Canonic Signed-Digit (CSD) system provides a unique representation without consecutive nonzero digits. The complex arithmetic proposed by Pekmestzi, is a starting point from which we may obtain a signed-digit representation of Gaussian integers. Since, in CSD arithmetic, the set of digits is extended by the digit ‘-1’, a natural generalization of the Pekmestzi arithmetic is the following:

$$z = \sum_j d_j 2^j \quad d_j \in \{0, 1, i, 1+i, 1-i, -i, -1+i, -1, -1-i\} \quad (3)$$

Clearly, this representation is redundant; however, as we will argue, it is not trivial to define a minimal (or canonic) representation.

A most straightforward technique for finding some representation of  $z = x + yi$  in the form of eqn. (3) is to find CSD representations of  $x$  and  $y$  and to combine them; that is, if  $a$  and  $b$  are the corresponding signed-digits (0,1 or -1) of  $x$  and  $y$ , respectively, then the corresponding digit of  $z$  is  $a + bi$ . However, as we see in the following example, the direct combination of the minimal real forms does not guarantee minimality (in terms of nonzero digits) of the representation.

**Example:** Let  $z = 10 + 5i$ . The CSD representations of 10 and 5 are  $(1010)_2$  and  $(0101)_2$  respectively, therefore the corresponding signed-digit representation of  $z$  is:

$z = 1 \cdot 2^3 + i \cdot 2^2 + 1 \cdot 2^1 + i \cdot 2^0$  which is not minimal, since we can represent  $z$  as:

$$z = (1 + i) \cdot 2^3 + 0 \cdot 2^2 + (1 - i) \cdot 2^1 + (-i) \cdot 2^0$$

The existence of reduction rules, such as:

$$(1, i, 1, i) \rightarrow (1 + i, 0, 1 - i, -i) \quad (4)$$

where  $\rightarrow$  represents the reduction rule mapping, is a key point in finding faster algorithms for multi-exponentiations based on this arithmetic.

The above example demonstrates that finding a minimal representation of a complex number is not a trivial task; however, a representation based on a single combination of CSD representations is a good starting point for finding more sparse representations (i.e. representations with smaller numbers of non-zero digits).

The reduction rule of eqn. (4) is an example of reducing 4 consecutive nonzero complex digits to 3. There are several combinations of two successive non-zeros reducing to one non-zero, but they cannot arise from combining canonic SD representations of the real and imaginary parts.

In the following we analyze the possibility of reducing 3 consecutive nonzero digits to 2; the total number of possible triples of complex digits obtained in this way is 121.

The following theorem, using the explicit knowledge that no consecutive non-zero digits occur in a CSD representation, provides the total number of Gaussian integer CSD  $n$ -tuples for a given  $n$ .

**Theorem 1:** Let  $v = (v_1, v_2, \dots, v_n)$  be a sequence of Gaussian integers such that  $|Re(v_k), Im(v_k)| \leq 1$  for  $k = 1, \dots, n$  and let  $Re(v_k) \cdot Re(v_{k+1}) = 0$  and  $Im(v_k) \cdot Im(v_{k+1}) = 0$  for  $k = 1, \dots, n-1$ . Then there exist exactly

$$T_n = \left\lceil \frac{2^{n+2} + (-1)^{n+1}}{3} \right\rceil^2 \text{ such sequences for every } n \geq 1.$$

**Proof:** See Appendix. □

An analysis of the complete set of triples uncovers the existence of 8 possible ways to reduce a combination of three nonzero digits to two, namely those given in Table 2.

**Table 2. The 8 reduction rules**

Combination	Reduction
$i, 1, -i$	$0, 1+i, i$
$-i, 1, i$	$0, 1-i, -i$
$i, -1, -i$	$0, -1+i, i$
$-i, -1, i$	$0, -1-i, -i$
$1, i, -1$	$0, 1+i, 1$
$-1, i, 1$	$0, -1+i, -1$
$1, -i, -1$	$0, 1-i, 1$
$-1, -i, 1$	$0, -1-i, -1$

We performed computer simulations with 10,000 randomly chosen 512-bit integers. It was observed that in about 7% of the cases there existed the possibility of reducing three nonzero consecutive complex digits to two. This confirms the heuristics based on probabilistic considerations.

It is always possible to analyze more complicated reductions; however, we will restrict ourselves to reductions appropriate for the multi-exponentiation algorithm.

#### 4 Multi-exponentiation based on the new arithmetic

We now apply the above arithmetic for computation of  $z = x^m y^n \pmod{p}$ . As we have already mentioned in the introduction, we will only consider the cases when  $x^{-1} \pmod{p}$  and  $y^{-1} \pmod{p}$  are ‘easy’ operations. These correspond to one of the following situations:

1.  $x$  and  $y$  are constants; therefore  $x^{-1} \pmod{p}$  and  $y^{-1} \pmod{p}$  can be precomputed in advance. Note that this is exactly the case we have in signature verification using DSS.
2. When the operation is executed over an elliptic curve, where the division is replaced by subtraction.

## 4.1 Algorithm 2

Let us consider the following algorithm:

*Input:*  $m, n, x, y$  and  $p$ : positive integers,  $p$ -prime;

*Output:*  $z = x^m y^n \pmod{p}$

**Step 1:** Precompute  $a_1 = xy \pmod{p}$ ;  $a_2 = x^{-1} \pmod{p}$ ;  $a_3 = y^{-1} \pmod{p}$ ;  
 $a_4 = x^{-1}y \pmod{p}$ ;  $a_5 = xy^{-1} \pmod{p}$ ;  $a_6 = x^{-1}y^{-1} \pmod{p}$ .

**Step 2:** Find the canonic signed-digit representation of  $m$  and  $n$ :  $m = \sum_{j=0}^M m_j 2^j$  and

$$n = \sum_{j=0}^N n_j 2^j, (m_j, n_j) \in \{0, 1, -1\};$$

**Step 3:** Obtain a complex-digit representation of  $z = m + ni$ :  $z = \sum_{j=0}^{\max(M, N)} z_j 2^j$ , where

$$z_j = m_j + n_j i;$$

**Step 4:** set  $s=1$ ;  $j=\max(M, N)$

**Step 5:** if( $z_j == 1$ )  $s = s*x \pmod{p}$   
 if( $z_j == i$ )  $s = s*y \pmod{p}$   
 if( $z_j == 1+i$ )  $s = s*a_1 \pmod{p}$   
 if( $z_j == -1$ )  $s = s*a_2 \pmod{p}$   
 if( $z_j == -i$ )  $s = s*a_3 \pmod{p}$   
 if( $z_j == -1+i$ )  $s = s*a_4 \pmod{p}$   
 if( $z_j == 1-i$ )  $s = s*a_5 \pmod{p}$   
 if( $z_j == -1-i$ )  $s = s*a_6 \pmod{p}$   
 if ( $j>0$ )  $s=s*s \pmod{p}$ ;

**Step 6:**  $j=j-1$

if ( $j > 0$ ) go to Step 5  
 Otherwise: Output  $s$ .

Now let us analyze the average and worst case behavior of this algorithm. We will make a simplifying assumption of a uniform distribution of complex digits in order to provide a concise, if slightly inaccurate, analysis. The fact that the distribution is not uniform is due to the fact that some CSD representations do not exist (e.g.  $(-1, 0, -1)$ ) and so the mapped complex digit  $(-1, 0, -1-i)$  also does not exist. We still have the possibility for reduction of the nonzero digits, so the estimations obtained may not be the best possible, and the application of further nonzero digits reduction rules (such as  $(1, i, 1, i) \rightarrow (1+i, 0, 1-i, -i)$ ) will slightly decrease the multiplicative constant. Assuming a uniform distribution of the digits allows us to demonstrate that the difficult-to-obtain canonic form will not lead to major computational savings. Such an assumption is not precise, since the triples of digits in the CSD representation of reals are not equally probable. Also, from a practical view point, the application of all possible reduction rules will take exponential time, which is impractical, and the further reduction obtained will be marginal at best.

Since the average proportion of zeros in the CSD representation of real integers is  $2/3$ ; therefore the probability that a randomly chosen complex digit,  $z = m + ni$ , is zero is  $4/9$  (based on the assumption of uniform distribution). The average number of modular multiplications in this case is  $(14/9)\lfloor \log(\max(m, n)) \rfloor$  which is considerably better than  $1.75\lfloor \log(\max(m, n)) \rfloor$ . We can obtain more savings using our representation as triples, and it is likely that we can further reduce the nonzero digits by the application of reductions such as  $(1, i, 1, i) \rightarrow (1+i, 0, 1-i, -i)$ . Such extra steps will slightly decrease the multiplicative constant, but this comes at the price of very complex preprocessing of the exponents, which requires exponential time. This is to be compared with the application of the rules given in Step 5 of Algorithm 2 which require only linear time.

## 4.2 An Example

Consider the evaluation of  $z = x^{8912} y^{9445} \pmod{p}$ . The representation of the number  $d = 8912 + 9445i$  in Pekmestzi arithmetic is:

$$d = (1 + i, 0, 0, i, 1, 0, 1 + i, 1 + i, i, 1, 0, i, 0, i).$$

This representation corresponds to the following vector addition chain for the computation of  $z$ :

(1,1),(2,2),(4,4),(8,8),(8,9),(16,18),(17,18),(34,36)  
 (68,72),(69,73),(138,146),(139,147),(278,294)  
 (278,295),(556,590),(557,590),(1114,1180)  
 (2228,2360),(2228,2361),(4456,4722),(7912,9444)  
 (8912,9445),

that is, we have 21 modular multiplications. Representations of  $d$ , as previously defined are:

$$d = (1 + i, 0, 0, 1 + i, 0, -1 + i, 0, -1, -i, 1, 0, i, 0, i) \quad (5)$$

(based on the CSD representation of the real and imaginary parts) and

$$d = (1 + i, 0, 0, 1 + i, 0, -1 + i, 0, 0, -1 - i, -1, 0, i, 0, i) \quad (6)$$

(based on eqn. (5) plus the reduction rules of Table 2).

As we can see, the number of nonzero digits decreases, and so the corresponding vector addition-subtraction chains will be shorter:

(1,1),(2,2),(4,4),(8,8),(9,9),(18,18),(36,36),(35,37)  
 (70,74),(140,148),(139,148),(278,296),(278,295)  
 (556,590),(557,590),(1114,1180),(2228,2360)  
 (2228,2361), (4456,4722),(8912,9444),(8912,9445)

based on eqn. (5) (20 modular multiplications) and:

(1,1),(2,2),(4,4),(8,8),(9,9),(18,18),(36,36),(35,37)  
 (70,74),(140,148),(280,296),(279,295),(558,590)  
 (557,590),(1114,1180),(2228,2360),(2228,2361) (4456,4722),(8912,9444),(8912,9445)

based on eqn. (6) (19 modular multiplications).

Let us compare the proposed technique with published algorithms. In all cases the expected number of modular multiplications is  $\alpha \lfloor \log(\max(m, n)) \rfloor$ , therefore the major figure of merit is the constant  $\alpha$ . In Shamir's method [3] it is 1.75. If one uses the method proposed by Yen, Lai and Lenstra (sliding window) [30] the value of  $\alpha$  varies for the different window sizes. In our technique the window size is 1. Table 3 shows the values of  $\alpha$  for different approaches.

Our new technique can be easily combined with the ‘sliding window’ technique at the price of more precomputed values<sup>1</sup>.

**Table 3. A comparison between different algorithms**

<b>Algorithms for multi-exponentiation</b>	<b>Values of <math>\alpha</math></b>
Separate exponentiations	2.0
Shamir’s method	1.75
Yen-Laih-Lenstra (window size = 1)	1.75
Yen-Laih-Lenstra (window size = 2)	1.625
Algorithm-I	1.75
Algorithm-II (without Table 2 reductions)	1.556
Algorithm-II (with Table 2 reductions)	1.534

## 5 Comments and conclusions

In this paper we have presented new algorithms for modular exponentiations based on complex arithmetic representations. The applicability of the proposed algorithms based on signed-digit complex arithmetic crucially depends on the assumption that the inverse elements of  $x$  and  $y$  can be precomputed in advance. Let us comment on this.

The general formulation of the problem for evaluation modular exponentiations is: given  $x$ ,  $y$  and  $z$  (or  $x_i$ ,  $r_i$  and  $z$  in the case of multi-exponentiation), compute  $x^y \pmod{z}$  (or

$\prod_i x_i^{r_i} \pmod{z}$  respectively). In cryptographic algorithms, however, some of the parameters

are constants (the keys) and some of the parameters (those forming the message) are variables.

In the RSA cryptosystem, say, the major operation is a single exponentiation  $C = x^y \pmod{z}$ , where  $y$  and  $z$  are constants and  $x$  is the message. If one uses an addition-subtraction chain for

computing  $C$ , then the time needed to evaluate  $x^{-1} \pmod{z}$  must be included in the total estimation

of the complexity of the algorithm. The computation of  $x^{-1} \pmod{z}$  is usually performed using the Euclidean algorithm for computing the greatest common divisor, or, equivalently,

---

1. The same can be achieved by using higher radices than two. We are grateful to P. Montgomery for pointing this out during the ARITH13 presentation.

via the continued fraction expansion of  $\frac{x}{z}$ . The computational complexity of this problem has been a subject of considerable analysis and it is well-known that the average number of divisions necessary to evaluate  $x^{-1} \pmod{z}$  is  $\frac{12 \ln 2}{\pi^2} \ln z + 0.06$  [12]. The most precise estimation is due to Norton [36]. For uniformly distributed integers in the interval  $[1, N]$  and any positive real  $\epsilon$ , the Euclidean algorithm requires an average of:

$$\frac{12 \ln 2}{\pi^2} \left( \ln N - \frac{1}{2} + \frac{\xi'(2)}{\xi(2)} \right) + T - \frac{1}{2} + O\left(N^{\epsilon - \frac{1}{6}}\right)$$

steps, where  $\xi(x)$  is the Riemann  $\xi$ -function and  $T$  is Porters constant. In discrete-log based cryptosystems, typically  $x$  and  $z$  are constants, therefore the computation of  $x^{-1} \pmod{z}$  can be performed in advance and excluded from the total run-time estimation. These considerations are not necessary when one deals with elliptic-curve cryptosystems, because the division is replaced by a subtraction. An addition (subtraction) formula on elliptic curves does not contain a modular division particularly when homogeneous coordinates are used [19].

The technique proposed in this paper is based on signed-digit complex arithmetic. It is mainly oriented towards an efficient implementation of the Digital Signature Standard where it is possible to precompute the modular inversion of the public and private keys. Our method seems inappropriate to cases where this is not possible. As we have already pointed out, the canonic signed-digit representation of the real and imaginary parts of a given Gaussian integer does not guarantee a canonic signed-digit complex representation. Therefore, in order to obtain an optimal arithmetic algorithm for modular exponentiation, one has to find an algorithm for determination of the canonic complex-digit representation for Gaussian integers. Applying a succession of all possible reduction rules will reach the minimal representation; however, it needs exponentially many operations. Hence, the polynomial-time determination of a CSD complex representation is an important open problem.

The generalization of the proposed arithmetic to spaces with more dimensions is another avenue for possible improvements. The main results on this subject are still under investigation.

## 6 References

- [1] R.A. Rivest, A. Shamir and L. Adleman. "A method for obtaining digital signatures and

- public-key cryptosystems”, *Communications of the ACM*, vol.21, 1978, pp. 120-126
- [2] V. McLellan, “Password security-crypto in your VAX”, *Digital Review*, October 1986, 86, p. 86
- [3] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms”, *IEEE Trans.on Information Theory*, vol.31, 1985, pp. 469-472
- [4] E.F. Brickell and K.S. McCurley, “Interactive identification and digital signatures”, *AT&T Technical Journal*, 1991, pp. 74-86
- [5] C.P. Schnorr, “Efficient identification and signatures for smart cards”, in *Advances in Cryptology, Crypto’89, Lecture Notes in Computer Science*, vol.435, pp. 239-252
- [6] P. Downey, B. Leony and P. Sethi, “Computing sequences with addition chains”, *SIAM Journal on Computing*, vol.11, 1981, pp. 638-696
- [7] P.E. Valskii, “On the lower bounds of multiplications in evaluation of powers”, *Problems of Cybernetics*, vol.2, 1959, pp. 73-74 (in Russian)
- [8] A. Yao, ”On the power evaluation”, *SIAM Jour. on Computing*, vol.5,1976, pp. 100-103
- [9] N. Pippenger, “On the evaluation of powers and monomials”, *SIAM Journal on Computing*, vol.9, 1980, pp.230-250
- [10] Y. Yacobi, “Exponentiating faster with addition chains”, *Proc.of EUROCRYPT’90, 1990, Lecture Notes in Computer Science*, vol.473, pp. 222-229
- [11] O. Kochergin, “The evaluation of powers”, *Mathematical Problems of Cybernetics*, vol.26, 1995, pp. 76-82 (in Russian)
- [12] D.E. Knuth, ”Seminumerical algorithms”, *The Art of Computer Programming*, vol.2, Addison Wesley, 1969
- [13] J. Jedwab and C.J. Mitchel, “Minimum weight modified signed-digit representation and fast exponentiation”, *Electronics Letters*, vol.25, 1989, pp.1171-1172
- [14] C.N. Zhang, “An improved binary algorithm for RSA”, *Computers and Mathematics with Applications*, vol.25, 1993, pp. 15-24
- [15] V. Dimitrov and T. Cooklev, “Two algorithms for modular exponentiations using non-standard arithmetics”, *IEICE Trans. on Fundamentals*, vol.E78-A, 1995, pp. 82-87
- [16] C.-Y. Chen, C.-C. Chang and W.-P. Yang, “Hybrid method for modular exponentiation with precomputation”, *Electronics Letters*, vol.32, 1996, pp. 540-541
- [17] G.W. Reitweisner, “Binary arithmetics”, *Adv. in Computers*, vol.1,1960, pp. 231-308
- [18] S. Arno and F. Wheeler, “Signed-digit representations of minimal Hamming weight”, *IEEE Trans. on Computers*, vol.42, 1993, pp. 1007-1010
- [19] K. Koyama and Y. Tsuruoka, “A signed binary window method for fast computing over elliptic curves”, *IEICE Transactions on Fundamentals*, vol. E76-A, 1993, pp. 55-62
- [20] E.F. Brickell, “A fast modular multiplication algorithm with applications to two key cryptography”, *Proc. CRYPTO’82, Springer-Verlag*, 1982, pp. 450-456
- [21] F. Morain and J. Olivos, “Speeding up the computations on and elliptic curve using addi-

- tion-subtraction chain”, *Theor. Informatics and Applications*, vol.24, 1990, pp. 531-544.
- [22] K. Koyama, U. Maurer, T. Okamoto and S. Vanstone, “New public-key schemes on elliptic curves over the ring  $\mathbf{Z}_n$ ”, *Proc. CRYPTO’91*, Springer-Verlag, 1991, pp. 305-311
- [23] N. Koblitz, “A course in number theory and cryptography”, Springer-Verlag, 1987
- [24] D.M.Gordon, “A survey of fast exponentiation methods”, *Journal of Algorithms*, vol. 27, No. 1, April 1998, pp.129-146
- [25] “NIST: A proposed federal information processing standard for digital signatures standard (DSS)”, *Federal Register* 1991, vol.56, pp. 42980-42982
- [26] E.G. Strauss, “Addition chains of vectors”, *American Mathematical Monthly*, vol.71, 1964, pp. 807-808
- [27] H. Chabanne, “Factoring of  $x^n - 1$  and orthogonalization over finite fields of characteristic 2”, *Applicable Algebra in Engineering, Communications and Computing*, vol.6, 1995, pp. 57-63
- [28] A. Thiong Ly, “A deterministic algorithms for factorizing polynomials over extensions  $GF(p^n)$  of  $GF(p)$ ,  $p$  a small prime”, *Journal of Information and Optimization Science*, vol.10, 1989, pp. 337-344
- [29] V.D. Blondel and J.N. Tsitsiklis, “When is a pair of matrices mortal”, *Information Processing Letters*, vol. 63, 1997, pp. 283-286,
- [30] S.-M. Yen, C.-S. Laih and A.K. Lenstra, “Multi-exponentiation”, *IEE Proc.-Computers and Digital Techniques*, vol.141, 1994, pp. 325-326
- [31] K. Pekmestzi, “Complex number multipliers”, *IEE Proc.-Computers and Digital Techniques*, vol.136, 1989, pp. 70-75
- [32] D.E. Knuth, “An imaginary number system”, *Communications of the ACM*, vol.2, 1960, pp.18-23
- [33] D.L. Dietmeyer, “Conversion from positive to negative and imaginary radix”, *IEEE Transactions on Electronic Computers*, vol.1, 1963, pp. 25-33.
- [34] D.A. Pospelov, “Fundamentals of computer arithmetics”, Moscow, Visshaja Shkola, 1970, (in Russian)
- [35] J. Durpat, Y. Herreros and S. Kla, “New redundant representation of complex numbers and vectors”, *Proc. the 10th IEEE Symposium on Computer Arithmetic*, IEEE Press, 1991, pp. 2-9
- [36] G.H. Norton, “On the asymptotic analysis of the Euclidean algorithm”, *Journal of Symbolic Computation*, 1990, vol.10, pp. 53-58
- [37] S.-M. Yen and C.-S. Laih, “Improved digital signature suitable for batch verification”, *IEEE Trans. Computers*, vol.44, No.7, 1995, pp. 957-959
- [38] S.-M. Yen and C.-S. Laih, “The fast cascade exponentiation algorithm and its applications on cryptography”, *Proc. of the AUSCRYPT’92*, Springer-Verlag, 1992, pp. 10.20-10.24

- [39] Y. Tsuruoka and K. Koyama, “Fast exponentiation algorithms based on batch-processing and precomputation”, IEICE Trans. on Fundamentals, E80-A, No.1, 1997, pp. 34-39
- [40] D. Gordon, “Discrete logarithms in GF(P) using the number field sieve”, SIAM J.Discr. Math., vol.6, No.1, 1993, pp. 124-138
- [41] L. Adleman and J. Demarrais, “A subexponential algorithm for discrete logarithms over all finite fields”, Mathematics of Computation, vol.61, No.203, 1993, pp. 1-15.
- [42] A. Menezes and S. Vanstone, “Handbook of applied cryptography”, Academic Press, 1995

## 7 Acknowledgments

The authors would like to acknowledge financial assistance from the Natural Sciences and Engineering Research Council of Canada, the Micronet Network of Centres of Excellence and workstations and software from the Canadian Microelectronics Corporation.

## 8 Appendix

Let us denote as  $w_n^{(g)}$  the number of the sequences  $(v_1, v_2, \dots, v_n)$  satisfying the conditions in the theorem and such that the last element,  $v_n$ , is equal to  $g$ . Recall that  $g$  belongs to the set  $\{-1-i, -i, 1-i, -1, 0, 1, -1+i, i, 1+i\}$ . Then for the nine  $\{w_n^{(g)}\}$  we have the following recurrence relations:

$$\begin{aligned}
 w_n^{(0)} &= w_{n-1}^{(-1-i)} + w_{n-1}^{(-i)} + w_{n-1}^{(1-i)} + w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)} + w_{n-1}^{(-1+i)} + w_{n-1}^{(i)} + w_{n-1}^{(1+i)} \\
 w_n^{(-i)} &= w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)} \\
 w_n^{(i)} &= w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)} \\
 w_n^{(-1)} &= w_{n-1}^{(-i)} + w_{n-1}^{(0)} + w_{n-1}^{(i)} \\
 w_n^{(1)} &= w_{n-1}^{(-i)} + w_{n-1}^{(0)} + w_{n-1}^{(i)} \\
 w_n^{(-1-i)} &= w_{n-1}^{(0)} \\
 w_n^{(-1+i)} &= w_{n-1}^{(0)} \\
 w_n^{(1-i)} &= w_{n-1}^{(0)} \\
 w_n^{(1+i)} &= w_{n-1}^{(0)}
 \end{aligned}$$

Obviously  $w_n^{(-1-i)} = w_n^{(-1+i)} = w_n^{(1-i)} = w_n^{(1+i)} = z_n$  and

$w_n^{(-i)} = w_n^{(-i)} = w_n^{(-1)} = w_n^{(1)} = x_n$ . Denote  $w_n^{(0)}$  as  $y_n$ . Then for  $x_n$ ,  $y_n$  and  $z_n$  we have the following system of recurrence equations:

$$\begin{aligned}x_n &= 2x_{n-1} + y_{n-1} \\y_n &= 4x_{n-1} + y_{n-1} + 4z_{n-1} \\z_n &= y_{n-1}\end{aligned}$$

with initial conditions  $x_1 = y_1 = z_1 = 1$ . After some manipulations we obtain the following equation for  $x_n$ :

$$x_{n+1} = 3x_n + 6x_{n-1} - 8x_{n-2} \quad (7)$$

with initial conditions:  $x_1 = 1$ ,  $x_2 = 3$  and  $x_3 = 15$ .

The solution of eqn. (7) is:

$$\begin{aligned}x_n &= \frac{2 \cdot 4^n - (-2)^n - 1}{9}, \quad \text{and} \quad \text{therefore} \quad y_n = \frac{4^{n+1} + 4 \cdot (-2)^n + 1}{9} \quad \text{and} \\z_n &= \frac{4^n + 4 \cdot (-2)^{n-1} + 1}{9}.\end{aligned}$$

The number,  $T_n$ , of sequences we are looking for is equal to:

$$T_n = 4x_n + y_n + 4z_n = \left[ \frac{2^{n+2} + (-1)^{n+1}}{3} \right]^2$$

and so the theorem is proved. □

## Vassil S. Dimitrov

Vassil S. Dimitrov was born in Plovdiv, Bulgaria, in 1964. He received the M.Sc. degree in computer science from the Technical University of Sofia, Bulgaria and the Ph.D. in mathematics from the Mathematical Institute of the Bulgarian Academy of Sciences in 1995. From 1995 until 1997 Dr. Dimitrov was a postdoctoral fellow in the VLSI Research Group, University of Windsor, Canada. Since January 1998 Dr. Dimitrov has held the position of research scientist at Reliable Software Technologies Corporation. He is involved with DARPA sponsored projects on cryptanalysis

His research interests include number theoretic algorithms, computer arithmetic, cryptography, fast algorithms for digital signal processing and related topics.

Dr. Dimitrov is a member of the New York Academy of Sciences.

## Graham A. Jullien

Graham Jullien was educated in the United Kingdom, receiving degrees, in Electrical Engineering, from the Universities of Loughborough, Birmingham and Aston (Ph.D., 1969). He was a student engineer and data processing engineer at English Electric Computers, UK, from 1961 to 1966, and a visiting senior research engineer at the Central Research Laboratories of EMI Ltd., UK, from 1975 to 1976.

Since 1969 he has been with the Electrical Engineering Department of the University of Windsor, Ontario, Canada, and currently holds the rank of University Professor. He is also the Director of the VLSI Research Group at the University of Windsor. He was a member of the Board of Directors of the Canadian Microelectronics Corporation from 1990 to 1993 and is a Principle Researcher and member of the Coordinating Committee and Board of Directors of the Micronet Network of Centres of Excellence.

He has published widely in the fields of Computer Arithmetic, Digital Signal Processing and VLSI Systems, and teaches courses in related areas. He has served on the technical committees of many international conferences; he serves on the Editorial Board of the Journal of VLSI Signal Processing, and was an Associate Editor of the IEEE Transactions on Computers from 1994 to 1996. He hosted and was program co-chair of the 11th IEEE Symposium on Computer Arithmetic; he was also the program chair for the 8th Great Lakes Symposium on VLSI, and is the program chair for the 1999 Asilomar Conference on Signals, Systems and Computers.

## **W.C. Miller**

William C. Miller was born in Toronto, Ontario, Canada. He received the B.S.E. degree in Electrical Engineering from the University of Michigan, Ann Arbor, in 1960, and the M.A.Sc. and Ph.D. degree in Electrical Engineering from the University of Waterloo, Ontario in 1961 and 1969, respectively. He joined the Department of Electrical Engineering at the University of Windsor in 1968, where he currently holds the rank of Professor.

His research interests are oriented towards digital signal processing and the design of massively parallel VLSI processor architectures for application specific problems in the area of image processing relating to machine vision. He also teaches courses in the area of circuit theory, signal processing and system theory. Dr. Miller is also engaged extensively in industrial consulting work.

He was director of the CAD/CAM centre at the University of Windsor, for a two year period ending in 1988, and currently serves on the Board of Directors and Technical Advisory Committee of the Canadian Microelectronics Corporation. Dr. Miller is a registered Professional Engineer in the Province of Ontario, and a member of the IEEE.