Algorithms for multi-exponentiation based on complex arithmetic

V.S.Dimitrov, G.A.Jullien and W.C.Miller
VLSI Research Group, University of 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. The algorithms proposed are based on binary-like complex arithmetic, introduced by Pekmestzi and generalized in this paper.

1. Introduction

Most number-theoretic cryptosystem [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 $a_0=1,a_1,a_2,...,a_r=B$, where r is the number of additions and for all $i=1,2,...,r$, $a_i = a_j+a_k$ for some $k \le j < i$. Any addition chain corresponds to some algorithm for the evaluation of $A^B \pmod p$. The shortest addition chains produce optimal (in terms of the number of multiplications) algorithms for ME; 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 fairly small length is based on the binary decomposition of B. If $B = (j_{n-1}j_{n-2}\cdots j_1 j_0)_2$ then we have the following C-like pseudo-code algorithm:

```
C=1;
for(i=0; i<=n-1; i=i+1)
{
if(j[i] == 1) {C = (C*A) % N ; }
A = (A*A) % N;
}
```

Obviously the number of modular multiplications (MM) strongly depends on the Hamming weight (number of ones in the binary representation 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) representation 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:

$$B = \sum_{d=0}^{n-1} d_i 2^i \qquad d_i \in \{0, 1, -1\} \tag{1}$$

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 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.

Recently several cryptographic protocols using multiple modular exponentiations (or multi-exponentiations) have been proposed. The main operation here is the computation

1063-6889/97 $10.00 © 1997 IEEE

of $z = \prod_{i=1}^{k} x_i^{e_i} (\mathrm{mod}\ p)$, for $k>1$. Three of the currently proposed cryptographics protocols use $k=2$ [4][5][24]; 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 $z = x^m y^n (\mathrm{mod}\ p)$, assuming p to be prime. In terms of addition chains this can be associated with vector addition chains (VAC), as defined by Strauss [25].

It has been recognized [28] that the separate computation of powers $(z = (x^m (\mathrm{mod}\ p)) \cdot (y^n (\mathrm{mod}\ p)) \mathrm{mod}\ p)$ is not optimal even if one uses optimal algorithms for the computation of $x^m \mathrm{mod}\ p$ and $y^n \mathrm{mod}\ p$. A very simple example is the following: the computation of $z = x^2 y^2 (\mathrm{mod}\ p)$ based on separate multiplications needs 3 MMs, whereas the same can be computed as $z = (x \cdot y)^2 (\mathrm{mod}\ p)$ using only 2 MMs.

Some algorithms in computer algebra and coding theory also require at some stages the computation of multi-exponentiations [26][27].

This paper is organized as follows. Section 2 shows briefly the specifications for the digital signature standard (DSS) in order to justify the importance of the efficient computation of multi exponentiation. In Section 3 we define the problem of finding the VAC, in the language of Gaussian integers, using an efficient representation of complex numbers [29]. Section 4 introduces the use of signed-digit complex arithmetic. This number representation has some features that are significantly different from CSDBR. Section 5 demonstrates the computation of multi-exponentiations via signed-digit complex arithmetic. Section 6 concludes the paper.

2. The Digital Signature Standard (DSS)

DSS is the proposed Digital Signature Standard, which specifies a Digital Signature Algorithm (DSA) and is part of the U.S. government's Capstone project. It was selected by the National Institute of Standards and Technology [24], in cooperation with the National Security Agency, to be the digital authentication standard of the U.S.government; whether the government should in fact adopt it as the official standard is still under debate.

The DSA makes the use of the following parameters:

1. p is a prime modulus, where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64.
2. q is a prime divisor of p-1, where $2^{159} < q < 2^{160}$
3. $g = h^{(p-1)/q} \mathrm{mod}\, p$, where h is any integer with $1 < h < p-1$ such that $h^{(p-1)/q} \mathrm{mod}\, p > 1$ (g has order q mod p).
4. x is a randomly, or pseudorandomly, generated integer with $0 < x < q$
5. $y = g^x \mathrm{mod}\, p$
6. k is a randomly, or pseudorandomly, generated integer with $0 < k < q$

The integers p, q and g can be public and common to a group of users. A user's private and public keys are x and y, respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature.

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$$r = (g^k \mathrm{mod}\, p) \mathrm{mod}\, q \text{ and}$$

$$s = (k^{-1}(SHA(M) + xr)) \mathrm{mod}\, q$$

In the above, k^{-1} is the multiplicative inverse of k, mod q. The value of $SHA(M)$ is a 160-bit string output by the Secure Hash Algorithm (SHA), specified in Federal Information Processing Standards Publication 180.

The signature is transmitted along with the message to the verifier. Prior to verifying the signature in a signed message, p, q and g plus the sender's public key y and identity are made available to the verifier in a secure manner.

Let $\bar{M}$, $\bar{r}$ and $\bar{s}$ be the received versions of M, r and s, respectively. To verify the signature, the verifier checks to see that $0 < \bar{r} < q$ and $0 < \bar{s} < q$; if either condition is violated the signature is rejected. If these two conditions are satisfied, the verifier computes:

$$w = (\bar{s})^{-1} \mathrm{mod}\, q$$

$$u_1 = ((SHA(\bar{M}))w) \mathrm{mod}\, q$$

$$u_2 = (\bar{r}w) \mathrm{mod}\, q$$

$$v = (g^{u_1} y^{u_2} \mathrm{mod}\, p) \mathrm{mod}\, q$$

If $v = \bar{r}$, then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key x corresponding to y. If $v \neq \bar{r}$ the message should be considered invalid.

In the DSS system, signature generation is much faster than signature verification. The slowest operation is the computation of v, which requires multi exponentiation, and this drives the search for fast multi exponentiation algorithms.

3. 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 $z = x^m y^n (\mathrm{mod} p)$ 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 $\alpha_0 = (1,0)$, $\alpha_1 = (0,1)$, $\alpha_2 = (1,1),\ldots, \alpha_r = (m,n)$, 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 [29] introduced the following 'binary-like' representation of complex numbers:

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

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 [30]-[33]. The digit encoding [29] is shown in Table 1:

Table 1 Digit encoding of complex digits

Complex digit	Binary code
0	00
1	10
i	01
$1+i$	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 $s = (1+i, 0, 0, 1+i, 1+i, i, i, 1, 1, 1, 0, 1)$. If we use the above encoding scheme for the complex digits, we will have the following representation of s:

$$s = (1100001111010101011010100010).$$

3.1. Algorithm I

Using Pekmestzi's arithmetic we can proposed the following algorithm for computing $z = x^m y^n (\mathrm{mod} p)$:

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 $h = max(\lfloor \log m \rfloor, \lfloor \log n \rfloor)$. Denote this string as $H = (g_{2h-1} g_{2h-2} \cdots g_1 g_0)$.

Step 2: $z := 1$; $q = x*y \ (\mathrm{mod} p)$;

Step 3: Cycle through the elements of H looking at two consecutive elements simultaneously.

If $(g_k, g_{k-1}) = (1,0)$ then $z = z*x \bmod p$

If $(g_k, g_{k-1}) = (0,1)$ then $z = z*y \bmod p$

If $(g_k, g_{k-1}) = (1,1)$ then $z = z*q \bmod p$

$z = z*z \ (\mathrm{mod} p)$.

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 $x^{4893} (\mathrm{mod} p)$, $y^{5096} (\mathrm{mod} p)$ 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 $\frac{7}{4}h = \frac{7}{4}\lfloor \log(max(m,n)) \rfloor$.

4. A signed-digit complex arithmetic

The CSD allows a unique representation without consecutive nonzero digits. The complex arithmetic, proposed by Pekmestzi and used in the previous section, is a starting point from which we may obtain a signed-digit representation of Gaussian integers. In Pekmestzi arithmetic, the set of digits consists of all Gaussian integers having an absolute value smaller than 2 (0,1,i and $1+i$). Since, in the CSD binary arithmetic, the set of digits is extended by the digit '-1', a natural generalization of the Pekmestzi arithmetic is the following:

$$z = \sum_j d_j 2^j$$

$$d_j \in \{0, 1, i, 1+i, 1-i, -i, -1+i, -1, -1-i\} \qquad (3)$$

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 (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:

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

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. The reduction rule of eqn. (4) is an example of reducing 4 consecutive nonzero complex digits to 3. If we have a signed-digit complex representation based on combining the CSD representations of the real and imaginary parts of a given Gaussian integer, then obviously there is no possibility of reducing 2 consecutive nonzero digits to 1, and so we have to 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; a few examples are shown in Table 2:

Table 2 Examples of Gaussian integer CSD triples

-1-*i*,0,-1-*i*	-1-*i*,0,-1	-1-*i*,0,-1+*i*	-1-*i*,0,-*i*
-1-*i*,0,1	-1-*i*,0,1+*i*	-1,-*i*,-1	-1,-*i*,0
-1,0,-1+*i*	-1,0,-*i*	-1,0,0	-1,0,*i*
-1,*i*,-1	-1,*i*,0	-1,*i*,1	-1+*i*,0,-1-*i*
-1+*i*,0,0	-1+*i*,0,*i*	-1+*i*,0,1-*i*	-1+*i*,0,1
-*i*,-1,*i*	-*i*,0,-1-*i*	-*i*,0,-1	-*i*,0,-1+*i*
-*i*,0,1-*i*	-*i*,0,1	-*i*,0,1+*i*	-*i*,1,-ii
0,-1,-*i*	0,-1,0	0,-1,*i*	0,-1+*i*,0
0,0,-1-*i*	0,0,-1	0,0,-1+*i*	0,0,-*i*
0,0,1	0,0,1+i	0,i,-1	0,i,0
0,1,0	0,1,*i*	0,1+*i*,0	*i*,-1,-*i*
i,0,-1	*i*,0,-1+*i*	*i*,0,-*i*	*i*,0,0

The following theorem provides the total number of Gaussian integer CSD n-tuples for a given n.

Theorem 1: :Let $v = \{v_1, v_2, \ldots, v_n\}$ be a sequence of Gaussian integers such that $|Re(v_k), Im(v_k)| \le 1$ for $k = 1, \ldots, n$ and let $Re(v_k) \cdot Re(v_{k+1}) = 0$ and $Im(v_k) \cdot Im(v_{k+1}) = 0$ for $k = 1, \ldots, n-1$. Then there exist exactly $T_n = \left\lceil \frac{2^{n+2} + (-1)^{n+1}}{3} \right\rceil^2$ such sequences for every $n \ge 1$.

Proof: Let us denote as $w_n^{(g)}$ the number of the sequences $\{v_1, v_2, \ldots, v_n\}$ 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 $w_n^{(g)}$'s we have the following recurrence relations:

$$w_n^{(0)} = w_{n-1}^{(-1-i)} + w_{n-1}^{(-i)} + w_{n-1}^{(1-i)} + w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)} + w_{n-1}^{(-1+i)} + w_{n-1}^{(i)} + w_{n-1}^{(1+i)}$$

$$w_n^{(-i)} = w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)}$$

$$w_n^{(i)} = w_{n-1}^{(-1)} + w_{n-1}^{(0)} + w_{n-1}^{(1)}$$

$$w_n^{(-1)} = w_{n-1}^{(-i)} + w_{n-1}^{(0)} + w_{n-1}^{(i)}$$

$$w_n^{(1)} = w_{n-1}^{(-i)} + w_{n-1}^{(0)} + w_{n-1}^{(i)}$$

$$w_n^{(-1-i)} = w_{n-1}^{(0)}$$

$$w_n^{(-1+i)} = w_{n-1}^{(0)}$$

$$w_n^{(1-i)} = w_{n-1}^{(0)}$$

$$w_n^{(1+i)} = w_{n-1}^{(0)}$$

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:

$$x_n = 2x_{n-1} + y_{n-1}$$

$$y_n = 4x_{n-1} + y_{n-1} + 4z_{n-1}$$

$$z_n = y_{n-1}$$

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

$$x_{n+1} = 3x_n + 6x_{n-1} - 8_{n-2} \qquad (5)$$

with initial conditions: $x_1 = 1$, $x_2 = 3$ and $x_3 = 15$. The solution of eqn. (5) is:

$$x_n = \frac{2 \cdot 4^n - (-2)^n - 1}{9}, \text{ and therefore}$$

$$y_n = \frac{4^{n+1} + 4 \cdot (-2)^n + 1}{9} \text{ and}$$

$$z_n = \frac{4^n + 4 \cdot (-2)^{n-1} + 1}{9}.$$

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

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

so the theorem is proved. ❑

The first members of this sequence are: 9,25,121 (the case used in the paper), 441, 1849, 7225,...

Table 2 shows the distribution of the number of nonzero digits in the complete set of triples:

Table 2 Nonzero digit distribution of CSD triples

Number of nonzero digits in a triple	Occurrences
0	1
1	24
2	80
3	16

An analysis of these triples uncovers the existence of 8 possibilities to reduce a combination of three nonzero digits to two, namely those given in Table 3. It is always possible to analyze more complicated reductions; however, we will restrict ourselves to reductions appropriate for the multi-exponentiation algorithm.

Table 3 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

5. Multi-exponentiation based on the new arithmetic

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

1 x and y are constants; therefore $x^{-1}(\mathrm{mod} p)$ and $y^{-1}(\mathrm{mod} p)$ 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.

Let us consider the following algorithm:

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

Output: $z = x^m y^n (\mathrm{mod} p)$;

Step 1: Precompute $a_1 = xy(\mathrm{mod} p)$;

$a_2 = x^{-1}(\mathrm{mod} p)$; $a_3 = y^{-1}(\mathrm{mod} p)$;

$a_4 = x^{-1}y(\mathrm{mod} p)$; $a_5 = xy^{-1}(\mathrm{mod} p)$;

$a_6 = x^{-1}y^{-1}(\mathrm{mod} p)$;

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;

Step 5: Cycle through the complex digits, z_j, applying the following multiplication rules:

if(z_j == 1) $s = s*x$ (mod p)
if(z_j == i) $s = s*$y(mod p)
if(z_j == 1+i) $s = s*a_1$ (mod p)
if(z_j == -1) $s = s*a_2$ (mod p)
if(z_j == -i) $s = s*a_3$ (mod p)
if(z_j == -1+i) $s = s*a_4$ (mod p)
if(z_j == 1-i) $s = s*a_5$ (mod p)
if(z_j == -1-i) $s = s*a_6$ (mod p)

$s=s*s \pmod p$;

Step 6: Output s

Now let us analyze the average and worst case behavior of the proposed algorithm. As we have already pointed out, 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 $4/9$. The average number of modular multiplications in this case is $1.55555\ldots\lfloor \log(max(m,n)) \rfloor$ which is considerably better than $1.75\lfloor \log(max(m,n)) \rfloor$. We can obtain even more savings, however, using our representation as triples. From Table 2 it is clear that the probability that a randomly chosen triple consists of all zeros is $1/121$; to contain one nonzero digit is $24/121$; two nonzero digits is $80/121$ and three nonzero digits is $16/121$. In eight of the cases we have the possibility of reducing three nonzero digits to two (Table 3), and this reduces the average number of modular multiplications to $1.533516\ldots\lfloor \log(max(m,n)) \rfloor$.

We still have the possibility for reduction of the nonzero digits, so the estimations obtained may not be the best possible. The application of further nonzero digits reductions (such as $(1, i, 1, i) \rightarrow (1+i, 0, 1-i, -i)$) will slightly decrease the multiplicative constant; however, this is a marginal improvement for which the price of complex preprocessing of the exponents is too high. We can show this by using Theorem 1 and some probabilistic considerations. Analyzing the reduction of three consecutive nonzero complex digits to two, the only finally reducible combinations of complex digits are:

$$\ldots, \underbrace{\pm i, \pm 1, \pm i, \pm 1, \ldots, \pm i, \pm 1}_{k}, \ldots \qquad (6)$$

and

$$\ldots, \underbrace{\pm 1, \pm i, \pm 1, \pm i, \ldots, \pm 1, \pm i}_{k}, \ldots \qquad (7)$$

Therefore, to reduce k consecutive nonzero complex digits $(k \geq 4)$ to less, we have to check the succession of digits and apply the appropriate reduction rules. The number of combinations is 2^k. According to Theorem 1, we have $T_k = \left\lfloor \frac{2^{k+2} + (-1)^{k+1}}{3} \right\rfloor^2 \approx \frac{16}{9} \cdot 4^k$ possible combinations of complex digits. Therefore, the expected total reduction of the multiplicative constant, based on application of all possible reduction rules, will be less than $\delta = \sum_{k=4}^{\infty} \frac{2^k}{k \cdot \frac{16}{9} 4^k} = \frac{9}{16} \cdot \sum_{k=4}^{\infty} \frac{1}{k \cdot 2^k}$ in this case.

Evaluating the sum on the right hand side, yields:

$$\delta = \frac{9}{16} \cdot \left(\ln 2 - \frac{1}{2} - \frac{1}{8} - \frac{1}{24} \right) = 0.014895\ldots.$$

Therefore, the multiplicative constant can be reduced to no less than 1.518, at the price of very complex preprocessing of the exponents, which requires exponential time. On the other hand, the application of the rules given in Table 3 requires only linear time. If we consider, for example, 512-bit exponents, the applications of these rules will save, on average, 20 MMs at the price of one cycle of reducing, where possible, the triple complex digit. The complexity of this step is definitely less than 1 MM, and therefore it makes sense to apply this step before proceeding further. The following example is based on this consideration.

5.1. An Example

We apply the algorithms proposed to evaluate $z = x^{8912} y^{9445} \pmod p$. 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:

$$d = (1+i, 0, 0, 1+i, 0, -1+i, 0, -1, -i, 1, 0, i, 0, i) \qquad (8)$$

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

$$d = (1+i, 0, 0, 1+i, 0, -1+i, 0, 0, -1-i, -1, 0, i, 0, \textcircled{9}) \qquad (9)$$

(based on eqn. (8) plus the reduction rules of Table 3).

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. (8) (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. (9) (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 α. In Shamir's method (mentioned in [3]) it is 1.75. If one uses the method proposed by Yen, Laih and Lenstra (sliding window) [28] the value of α varies for the different window sizes. In our technique the window size is 1. Table 4 shows the values of α for different approaches.

Table 4 A comparison between different algorithms

Algorithms for multi-exponentiations	Values of α
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 3 reductions)	1.556
Algorithm-II (with Table 3 reductions)	1.534

The proposed technique can be easily combined with the 'sliding window' technique at the price of more precomputed values.

6. 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 $x^y(\mathrm{mod}z)$ (or $\prod_i x_i^{r_i}(\mathrm{mod}z)$ 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 $C = x^y(\mathrm{mod}z)$, 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 $x^{-1}\mathrm{mod}z$ must be included in the total estimation of the complexity of the algorithm. The computation of $x^{-1}\mathrm{mod}\ z$ is usually performed using the Euclidean algorithm for computing the greatest common divisor, or, equivalently, 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 $x^{-1}(\mathrm{mod}z)$ is $\frac{12\ln 2}{\pi^2}\ln z + 0.06$ [12]. The most precise estimation is due to Norton [34]. For uniformly distributed integers in the interval $[1, N]$ and any positive real ε, the Euclidean algorithm requires an average of $\frac{12\ln 2}{\pi^2}\left(\ln N - \frac{1}{2} + \frac{\xi'(2)}{\xi(2)}\right) + T - \frac{1}{2} + O\left(N^{\varepsilon - \frac{1}{6}}\right)$ steps, where $\xi(x)$ is the Riemann ξ-function and T is the Porters constant. In discrete-log based cryptosystems typically x and z are constants, therefore the computation of $x^{-1}\mathrm{mod}z$ can be performed in advance and excluded from the total runtime 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. 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.

7. 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, Oct.1986, 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 Journal 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 nonstandard arithmetics", IEICE Transactions 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", Advances 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, 1982, pp.450-456
[21] F.Morain and J.Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chain", Theoretical 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, 1991, pp.305-311
[23] N.Koblitz, "A course in number theory and cryptography", Berlin, Springer-Verlag, 1987
[24] "NIST: A proposed federal information processing standard for digital signatures standard (DSS)", Federal Register 1991, vol.56,pp.42980-42982
[25] E.G.Strauss, "Addition chains of vectors", American Mathematical Monthly,vol.71, 1964, pp.807-808
[26] 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
[27] 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
[28] S.-M.Yen,C.-S.Laih and A.K.Lenstra, "Multi-exponentiation", IEE Proc.-Computers and Digital Techniques,vol.141,1994, pp.325-326
[29] K.Pekmestzi, "Complex number multipliers", IEE Proc.-Computers and Digital Techniques,vol.136, 1989, pp.70-75
[30] D.E.Knuth, "An imaginary number system", Communications of the ACM, vol.2, 1960, pp.18-23
[31] D.L.Dietmeyer, "Conversion from positive to negative and imaginary radix", IEEE Transactions on electronic computers, vol.1,1963
[32] D.A.Pospelov, "Fundamentals of computer arithmetics", Moscow, Visshaja Shkola, 1970, (in Russian)
[33] J.Durpat,Y.Herreros and S.Kla, "New redundant representation of complex numbers and vectors", Proc. the 10th IEEE Symposium on computer arithmetic, 1991, pp.2-9
[34] G.H.Norton, "On the asymptotic analysis of the Euclidean algorithm", Journal of Symbolic Computation, 1990, vol.10, pp.53-58

8. 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.