Note

Exponentiation using canonical recoding

Ömer Eğecioğlu
Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

Çetin Kaya Koç*
Department of Electrical and Computer Engineering, Oregon State University, Corvallis, OR 97331, USA

Communicated by O.H. Ibarra
Received April 1993
Revised August 1993

Abstract


The canonical bit recoding technique can be used to reduce the average number of multiplications required to compute $X^E$ provided that $X^{-1}$ is supplied along with $X$. We model the generation of the digits of the canonical recoding $D$ of an $n$-bit long exponent $E$ as a Markov chain, and show that binary, quaternary, and octal methods applied to $D$ require $\frac{3}{2}n$, $\frac{5}{3}n$, and $\frac{7}{5}n$ multiplications, compared to $\frac{3}{2}n$, $\frac{5}{3}n$, and $\frac{7}{5}n$ required by these methods applied to $E$. We show that, in general, the canonically recoded $m$-ary method for constant $m$ requires fewer multiplications than the standard $m$-ary method. However, when $m$ is picked optimally for each method for a given $n$, then the average number of multiplications required by the standard method is fewer than those required by the recoded version.

1. Introduction

The binary method [5] computes $Y = X^E$ using $n-1$ squarings and as many multiplications as one less than the number of nonzero bits in the binary expansion of

Correspondence to: Ö. Eğecioğlu, Department of Computer Science, University of California, Santa Barbara, CA 93106, USA. E-mail: omer@cs.ucsb.edu.

*Supported in part by RSA Data Security Inc., Redwood City, CA, USA.
the exponent, where \( n = 1 + \lfloor \log_2 E \rfloor \). It is well known that \( n - 1 \) is a lower bound for the number of squaring operations required. However, it is possible to reduce the number of subsequent multiplications using a recoding of the exponent \([4, 6, 7, 10, 16]\). Recoding techniques (Booth recoding, bit-pair recoding, etc.) for sparse representations of binary numbers have been effectively used in multiplication algorithms \([3, 17]\). For example, the original Booth recoding technique \([1]\) scans the bits of the multiplier one bit at a time, and adds or subtracts the multiplicand to or from the partial product, depending on the value of the current bit and the previous bit. The modified versions of the Booth algorithm scan the bits of the multiplier two bits at a time \([9]\) or three bits at a time \([17]\). These techniques are equivalent in the sense that a signed-digit representation which is based on the identity \( 2^{i+j} - 2^i = 2^{i+j-1} + 2^{i+j-2} + \cdots + 2^{i+1} + 2^i \) is used to collapse blocks of 1's appearing in a binary representation. In a signed-digit number with radix 2, three symbols \( \{1, 0, 1\} \) are allowed for the digit set, in which 1 and \( \bar{1} \) in bit position \( i \) represent \( +2^i \) and \( -2^i \), respectively.

Bit recoding techniques applied to \( E \) can be used for the exponentiation problem provided that \( X^{-1} \) is supplied along with \( X \). Throughout this paper, we will ignore the preprocessing time required for the computation of \( X^{-1} \) and treat it as part of the input.

2. Canonical recoding

A signed-digit vector \( D \) of \( E \) is a sparse recoding of \( E \) using digits from the set \( \{1, 0, 1\} \). The recoding is canonical if \( D \) contains no adjacent nonzero digits \([3, 8, 13]\). Thus, a canonical signed-digit vector of \( E \) is of the form \( D = (D_{n-1} D_{n-2} \cdots D_0) \) with \( D_i \in \{1, 0, 1\} \) and

\[
D_i D_{i-1} = 0 \quad \text{for} \quad 1 \leq i \leq n-1.
\]

It can be shown that the canonical signed-digit vector for \( E \) is unique if the binary expansion of \( E \) is viewed as padded with an initial zero. This canonical signed-digit vector can be constructed by the canonical recoding algorithm of Reitwiesner \([13]\). Reitwiesner's algorithm computes \( D \) starting from the least significant digit and proceeding to the left. First the auxiliary carry variable \( C_0 \) is set to 0 and subsequently the binary expansion of \( E \) is scanned two bits at a time. The canonically recoded digit \( D_i \) and the next value of the auxiliary binary variable \( C_{i+1} \) for \( i = 0, 1, 2, \ldots, n \) are generated using Table 1. As an example, when \( E = 3038 \), we compute the canonical signed-digit vector \( D \) as

\[
E = (0101011110111110) = 2^{11} + 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1,
\]

\[
D = (1010000000010) = 2^{12} - 2^{10} - 2^8 - 2^7.
\]

Note that in this example the exponent \( E \) contains nine nonzero bits while its canonically recoded version contains only four nonzero digits. Consequently, the
binary method requires $11 + 8 = 19$ multiplications to compute $X^{3038}$ when applied to the binary expansion of $E$, but only $12 + 3 = 15$ multiplications when applied to the canonical signed-digit vector $D$. The canonical signed-digit vector $D$ is optimal in the sense that it has the minimum number of nonzero digits among all signed-digit vectors representing the same number.

3. The standard $m$-ary method

The binary method can be generalized to the (standard) $m$-ary method [5, 12, 16] which scans the digits of $E$ expressed in radix $m$. We restrict our attention to the case when $m = 2^d$ for some $d$. Let $E = (E_{d-1} E_{d-2} \cdots E_1 E_0)$ be the binary expansion of the exponent. We will assume that the most significant bit is equal to zero, i.e., $E_{d-1} = 0$. This representation of $E$ is partitioned into $k$ blocks of length $d$ each, for $kd = n$ (if $d$ does not divide $n$, the exponent is padded with at most $d - 1$ zeros). Define

$$F_i = (E_{id+d-1} E_{id+d-2} \cdots E_{id}) = \sum_{r=0}^{d-1} E_{id+r} 2^r.$$  

(1)

Note that $0 \leq F_i \leq 2^d - 1$ and $E = \sum_{i=0}^{k-1} F_i 2^d$. In the preprocessing phase of the $m$-ary method, the values of $X^F$ for $F = 2, 3, \ldots, 2^d - 1$ corresponding to all possible values of the length-$d$ bit-sections are computed. Next, the bits of $E$ are scanned $d$ bits at a time from the most significant to the least significant. At each step the partial result is raised to the $2^d$ power and multiplied with $X^{F_i}$, where $F_i$ is the (nonzero) value of the current bit section.

**Standard $m$-ary method**

*Input:* $X, E, n, d$ where $n = 1 + \lfloor \log_2 E \rfloor$ and $n = kd$ for $k \geq 1$.

*Output:* $Y = X^E$.

1. Decompose $E$ into $d$-bit words $F_i$ for $i = 0, 1, 2, \ldots, k-1$.
2. Compute and store $X^{F_i}$ for all $F = 2, 3, 4, \ldots, 2^d - 1$. 

**Table 1**

<table>
<thead>
<tr>
<th>$E_{i+1}$</th>
<th>$E_i$</th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$C_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
3. \( Y := X^F_{k-1} \)
4. for \( i = k - 2 \) downto 0
   4a. \( Y := Y^2^i \)
   4b. if \( F_i \neq 0 \) then \( Y := Y \cdot X^F_i \)
5. return \( Y \)

The preprocessing part in Step 2 of the \( m \)-ary method requires \( 2^d - 2 \) multiplications. The number of squaring operations in Step 4a is equal to \((k - 1)d\). Multiplications in Step 4b are performed for nonzero values of \( F_i \). Since \( m - 1 \) out of \( m \) possible values of \( F_i \) are nonzero, the average number of subsequent multiplications required is \((k - 1)((m - 1)/m)\). Thus, we find that, on the average, the \( m \)-ary method requires a total of

\[
T_i(n, d) = n - d + \left(\frac{n}{d} - 1\right)\left(1 - \frac{1}{2^d}\right) + 2^d - 2
\]  \( (2) \)

multiplications. The average number of squarings plus multiplications for the binary \((d = 1)\), the quaternary \((d = 2)\), and the octal \((d = 3)\) methods are found from (2) as

\[
T_i(n, 1) = \frac{3}{2} n - \frac{3}{2}, \quad T_i(n, 2) = \frac{3}{8} n - \frac{3}{4}, \quad T_i(n, 3) = \frac{3}{8} n - \frac{1}{8},
\]  \( (3) \)

respectively.

4. The recoded \( m \)-ary method

In the recoded \( m \)-ary method, we partition the canonical signed-digit vector \( D \) produced by Reitwiesner's algorithm instead of the exponent \( E \) itself. In other words, in the recoded \( m \)-ary method the \( d \)-bit-at-a-time partitioning that determines the bit sections \( F_i \) in (1) is applied to the canonical signed-digit vector \( D \).

Recoded \( m \)-ary method

Input: \( X, X^{-1}, E, n, d \) where \( n = 1 + \lceil \log_2 E \rceil \) and \( n = kd \) for \( k \geq 1 \).
Output: \( Y = X^E \).
1. Compute the canonical signed-digit recoding \( D \) of \( E \) using Reitwiesner's algorithm.
2. Decompose \( D \) into \( d \)-bit words \( F_i \) for \( i = 0, 1, 2, \ldots, k - 1 \).
3. Compute and store \( X^F_i \) for all possible length-\( d \) bit-sections that can appear in \( D \).
4. \( Y := X^F_{k-1} \)
5. for \( i = k - 2 \) downto 0
   5a. \( Y := Y^{2^i} \)
   5b. if \( F_i \neq 0 \) then \( Y := Y \cdot X^F_i \)
6. return \( Y \)
5. Analysis of the recoded $m$-ary method

In the analysis of the recoded $m$-ary method we need the number of all possible length-$d$ sections $F$ of a canonical signed-digit vector to estimate the work involved in Step 3 of the recoded $m$-ary method. To compute this, denote by $L$ the formal language of all words $w$ over the alphabet $\{1, 0, 1\}$ in which none of the patterns

$$11, \quad 1\bar{1}, \quad \bar{1}1, \quad \bar{1}\bar{1}$$

appears. Thus, the words $w$ of length $d$ in $L$ correspond to possible length-$d$ sections $F$ of a canonical signed-digit vector. For $d \geq 0$, let $\tau_d$ denote the total number of words of length $d$ in $L$. We have the following result.

**Lemma 5.1.**

$$\tau_d = \frac{1}{3} \left[ 2^{d+2} + (-1)^{d+1} \right]. \quad (4)$$

**Proof.** By considering the words in $L$ according to their first letter, we see

$$L = \lambda + 1 + \bar{1} + 10L + \bar{1}0L + 0L,$$

where $\lambda$ denotes the empty word and $+$ denotes disjoint union. Consider the generating function

$$f_L(t) = \sum_{w \in L} t^{|w|} = \sum_{d \geq 0} \tau_d t^d.$$

It follows from (5) that $f_L$ satisfies

$$f_L(t) = 1 + 2t + 2t^2 f_L(t) + tf_L(t),$$

and therefore

$$f_L(t) = \frac{1 + 2t}{1 - t - 2t^2} = \frac{4}{3} \frac{1}{1 - 2t} \frac{1}{3 + t}. \quad (6)$$

Now (4) follows by equating the coefficient of $t^d$ on both sides of (6). \qed

Since we are interested in the cost of multiplications required in exponentiation, we do not concern ourselves with the bit level preprocessing required for the computation of the canonical recoding $D$ in Step 1 of the algorithm. The number of multiplications necessary in the preprocessing stage Step 3 of the recoded $m$-ary method is given by the following lemma.

**Lemma 5.2.** The number of multiplications required in the preprocessing phase (Step 3) of the recoded $m$-ary method is

$$\tau_d - 3 = \frac{1}{3} \left[ 2^{d+2} + (-1)^{d+1} \right] - 3.$$
Proof. We need to compute the number of multiplications required to compute $X^F$ for all length-$d$ canonical signed-digit vectors $F$. First we compute all quantities $X^F$ where $F$ contains only one nonzero letter. Since $1$, $X$, and $X^{-1}$ are already available, this step requires $2(d-1)$ multiplications. After this, each value $X^F$, where $F$ contains $k > 1$ nonzero digits, can be computed recursively from the already computed values of $X^F$ where $F$ contains fewer than $k$ nonzero digits by a single multiplication. For $k \geq 0$, let $c_k$ denote the number of words of length $d$ in $\mathcal{L}$ with exactly $k$ nonzero letters. Then the total number of multiplications required for the pre-processing step is

$$2(d-1) + c_2 + c_3 + \cdots$$

Since $c_0 = 1$, $c_1 = 2d$ and

$$\tau_d = c_0 + c_1 + c_2 + \cdots,$$

it follows that the total number of multiplications required is $\tau_d - (1 + 2d) + 2(d-1) = \tau_d - 3$. □

Now we turn to the computation of the probability that a length-$d$ canonically recoded bit-section $F$ consists of $d$ zeros, since the multiplication in Step 5b of the recoded $n$-ary method is carried out only for nonzero $F$. An $n$-bit binary number $E$ uniformly distributed in the range $[0,2^n-1]$ can be viewed as the output of a random process that generates one bit at a time. Each bit assumes a value of zero or one with equal probability and there is no dependency between any two bits generated. Thus, $\mathcal{P}(E_i = 0) = \mathcal{P}(E_i = 1) = \frac{1}{2}$ for $0 \leq i \leq n-1$. The signed-digit numbers produced by the canonical recoding algorithm can be modeled using a finite Markov chain. The state variables are taken to be the triplets $(E_{i+1}, E_i, C_i)$. There are eight states for the eight possible combinations of input as given in Table 1. The state transitions given in Table 2 are produced by considering all eight states labeled $s_0$ to $s_7$ and their successors from Table 1. As an example, consider state $s_0$ which represents $(E_{i+1}, E_i, C_i) = (0,0,0)$. We compute the output $(D_i, C_{i+1})$ as $(0,0)$ from Table 1. Thus, the next state is $(E_{i+2}, E_{i+1}, C_{i+1}) = (E_{i+2}, 0, 0)$. Since $\mathcal{P}(E_{i+2} = 0) = \mathcal{P}(E_{i+2} = 1) = \frac{1}{2}$, there are transitions from state $s_0$ to the states $s_0 = (0,0,0)$ and $s_4 = (1,0,0)$, with equal probability.

Let $\mathcal{P}_{ij}$ denote the probability that the successor state of $s_i$ is $s_j$. From the above analysis $\mathcal{P}_{00} = \mathcal{P}_{04} = \frac{1}{2}$ and $\mathcal{P}_{ij} = 0$ for $j = 1,2,3,5,6,7$. After computing the probabilities $\mathcal{P}_{ij}$ for all $i$ and $j$ from Table 2, we find that the one-step transition probability
matrix of the chain is

\[
P = \begin{bmatrix}
1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\
0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\
0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\
\end{bmatrix}
\]  \quad (7)

The limiting probability \( \pi_i \) of state \( s_i \) can be found by solving the system of linear equations \( \pi P = \pi \) with \( \pi_0 + \pi_1 + \cdots + \pi_7 = 1 \). This gives

\[
\pi = [\frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{6}, \frac{1}{6}] .
\]  \quad (8)

Having computed \( \pi_i \) and \( \mathcal{P}_{ij} \) for all \( 0 \leq i, j \leq 7 \), we can easily prove several properties of the canonical recoding algorithm. For example, the probability that a digit in a canonical signed-digit number \( D \) is equal to zero is found by summing the limiting probabilities of the states for which output \( D_i = 0 \). From Table 2 and (8) we get

\[
\mathcal{P}(D_i = 0) = \pi_0 + \pi_3 + \pi_4 + \pi_7 = \frac{3}{2}.
\]

In particular, the average number of nonzero digits in the canonically recoded binary number \( D \) is equal to \( \frac{1}{3} n \). Therefore, the average number of squarings plus multiplications required by the recoded binary method for large \( n \) is \( \frac{3}{2} n + O(1) \), which is better than \( T_0(n, 1) = \frac{1}{2} n + O(1) \) required by the standard binary method.
Theorem 5.3. The probability that a length-$d$ bit-section $F$ in a canonically recoded signed-digit vector has all bits equal to zero is
\[
\binom{d-1}{2} \frac{2}{3} = \frac{1}{3 \cdot 2^{d-2}}.
\]

Proof. We have already seen that $\mathcal{P}(D_i = 0) = \frac{2}{3}$. The probability that $D_{i+1} = 0$ when $D_i = 0$ is found as
\[
\mathcal{P}(D_{i+1} = 0 | D_i = 0) = \frac{\sum_{j=0,3,4,7} \pi_0 \mathcal{P}_{0j} + \pi_3 \mathcal{P}_{3j} + \pi_4 \mathcal{P}_{4j} + \pi_7 \mathcal{P}_{7j}}{\pi_0 + \pi_3 + \pi_4 + \pi_7} = \frac{1}{2}.
\]

It follows that for $d \leq 1$
\[
\mathcal{P}(D_{i+d-1} = 0 | D_{i+d-2} = 0, D_{i+d-3} = 0, \ldots, D_i = 0) = \left(\frac{1}{2}\right)^{d-1} \frac{2}{3}. \quad \Box
\]

Combining Theorem 5.3 and the preprocessing cost for the recoded $m$-ary method given in Lemma 5.2, we find that, on the average, the recoded $m$-ary method with $m = 2^d$ requires a total of
\[
T_i(n, d) = n - d + \left(1 - \frac{1}{3 \cdot 2^{d-2}}\right) \left(\frac{n}{d} - 1\right) + \frac{1}{3} \left[2^{d+2} + (-1)^{d+1}\right] - 3 \quad (9)
\]
squarings and multiplications. Figure 1 compares the average number of multiplications required by the standard and the recoded $m$-ary methods, respectively, as a function of $n = 2^7, 2^8, \ldots, 2^{16}$ and $d = 1, 2, \ldots, 15$.

The average number of squarings plus multiplications for the recoded binary ($d = 1$), the recoded quaternary ($d = 2$), and the recoded octal ($d = 3$) methods are found from (9) as
\[
T_i(n, 1) = \frac{2}{3} n - \frac{4}{3}, \quad T_i(n, 2) = \frac{3}{2} n - \frac{3}{2}, \quad T_i(n, 3) = \frac{7}{18} n + \frac{75}{18}, \quad (10)
\]
respectively.

6. Comparison of standard and recoded $m$-ary methods

For large $n$ and fixed $d$, the behavior of $T_i(n, d)$ given in (9) and $T_i(n, d)$ of the standard $m$-ary method given in (2) is governed by the coefficient of $n$. In Table 3 we compare the values $T_i(n, d)/n$ and $T_i(n, d)/n$ for large $n$.

We can compute directly from the expressions in (2) and (9) that for constant $d$
\[
\lim_{n \to \infty} \frac{T_i(n, d)}{T_i(n, d)} = \frac{(d+1)2^d - \frac{4}{3}}{(d+1)2^d - 1} < 1. \quad (11)
\]
Table 3
The average number of multiplications for the recoded and standard m-ary methods

<table>
<thead>
<tr>
<th>d = log₂ n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_s(n, d)/n</td>
<td>1.5</td>
<td>1.375</td>
<td>1.29167</td>
<td>1.23437</td>
<td>1.19375</td>
<td>1.16406</td>
<td>1.14174</td>
<td>1.12451</td>
</tr>
<tr>
<td>T_r(n, d)/n</td>
<td>1.33333</td>
<td>1.33333</td>
<td>1.27778</td>
<td>1.22917</td>
<td>1.19167</td>
<td>1.16319</td>
<td>1.14137</td>
<td>1.12435</td>
</tr>
</tbody>
</table>

It is interesting to note that if we consider the optimal values $d_s$ and $d_r$ of $d$ (which depend on $n$) which minimize the average number of multiplications required by the standard and the recoded m-ary methods, respectively, then

$$\frac{T_r(n, d_r)}{T_s(n, d_s)} > 1$$

(12)

for large $n$. To prove (12), we consider the behavior of $T_s(n, d)$ and $T_r(n, d)$ for large $n$ and ignore the lower-order terms involving $d$ in (2) and (9). By differentiation, the optimal values $d = d_s$ and $d = d_r$ of the lengths of the bit-sections in the standard and
the recoded $m$-ary methods that minimize the number of multiplications are found to be

$$\frac{d^2 2^{2d} \log 2 - d^2 \log 2}{2^d - d \log 2 - 1} = n \quad \text{and} \quad \frac{4d^2 2^{2d} \log 2 - 4d^2 \log 2}{3 2^d - 4d \log 2 - 4} = n,$$

respectively. Since $d$ increases without bound in each of these equations as $n$ gets large, $d_s$ and $d_1$ satisfy

$$d_s^2 2^{d_s} \log 2 \approx n \quad \text{and} \quad d_1^2 2^{d_1} \log 2 \approx \frac{3}{2} n.$$

The function $d^2 2^{d}$ is an increasing function of $d$ and therefore $d_s < d_1$. Now from the expressions in (2) and (9) we get

$$\frac{T_s(n, d_s)}{T_s(n, d_1)} \approx 1 + 1/d_s,$$

for large $n$, which implies (12). Exact values of $d_s$ and $d_1$ for a given $n$ can be obtained by enumeration. These optimal values of $d_s$ and $d_1$ are given in Table 4 together with the corresponding values of $T_s$ and $T_1$ for each $n = 2^7, 2^8, \ldots, 2^{16}$.

7. Remarks

Algorithms for computing $X^E$ using as few multiplications as possible are crucial in many important applications in computer science and engineering. Recent applications in cryptography, for example, the RSA algorithm [14], the ElGamal signature scheme [2], and the recently proposed digital signature standard (DSS) of National Institute for Standards and Technology [11], require the computation of $X^E \pmod{M}$ for large values of $E$ (usually $n = \log_2 E \geq 512$). The recoded $m$-ary method can be useful for these particular applications if $X^{-1}$ can be supplied without too much extra cost. Even though the inverse $X^{-1} \pmod{M}$ can easily be computed using the

<table>
<thead>
<tr>
<th>$n$</th>
<th>$d_s$</th>
<th>$T_s(n, d_s)$</th>
<th>$d_1$</th>
<th>$T_s(n, d_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>4</td>
<td>168</td>
<td>3</td>
<td>168</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>326</td>
<td>4</td>
<td>328</td>
</tr>
<tr>
<td>512</td>
<td>5</td>
<td>636</td>
<td>4</td>
<td>643</td>
</tr>
<tr>
<td>1024</td>
<td>5</td>
<td>1247</td>
<td>5</td>
<td>1255</td>
</tr>
<tr>
<td>2048</td>
<td>6</td>
<td>2440</td>
<td>6</td>
<td>2458</td>
</tr>
<tr>
<td>4096</td>
<td>7</td>
<td>4795</td>
<td>7</td>
<td>4836</td>
</tr>
<tr>
<td>8192</td>
<td>8</td>
<td>9457</td>
<td>7</td>
<td>9511</td>
</tr>
<tr>
<td>16384</td>
<td>8</td>
<td>18669</td>
<td>8</td>
<td>18751</td>
</tr>
<tr>
<td>32768</td>
<td>9</td>
<td>36902</td>
<td>9</td>
<td>37070</td>
</tr>
<tr>
<td>65536</td>
<td>10</td>
<td>73095</td>
<td>10</td>
<td>73433</td>
</tr>
</tbody>
</table>
extended Euclid algorithm, the cost of this computation far exceeds the time gained by
the use of the reoding technique in exponentiation. Thus, at this time the reoding
techniques do not seem to be particularly applicable to these cryptosystems. However,
the reoding techniques may be useful for computations on elliptic curves over finite
fields, since in these cases the inverse is available at no additional cost [6, 10]. In this
context, one computes $E \cdot X$, where $E$ is a large integer and $X$ is a point on the elliptic
curve. The multiplication operator is determined by the group law of the elliptic curve.
An algorithm for computing $X^E$ is easily converted to an algorithm for computing
$E \cdot X$, where we replace multiplication by addition and division (multiplication with
the inverse) by subtraction.

References

(also reprinted in [15, pp. 100–104]).
Cryptography – Proc. CRYPTO ’89, Lecture Notes in Computer Science, Vol. 576 (Springer, New York,
reprinted in [15, pp. 14–38]).
[10] F. Morain and J. Olivos, Speeding up the computations on an elliptic curve using addition–subtraction
[12] H. Orop and P. Komerup, A high-radix hardware algorithm for calculating the exponential $M^E$
1990).
[16] N. Takagi, A radix-4 modular multiplication hardware algorithm for modular exponentiation, IEEE
York, 1982).