Exponentiation using Canonical Recoding

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

Çetin Kaya Koç*
Department of Electrical & Computer Engineering
Oregon State University
Corvallis, Oregon 97331

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 $\frac{4}{3}\,n$, $\frac{4}{3}\,n$, and $\frac{23}{18}\,n$ multiplications, compared to $\frac{3}{2}\,n$, $\frac{11}{8}\,n$, and $\frac{31}{24}n$ 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.

Key Words: Exponentiation, binary method, m-ary method, canonical recoding.

1 Introduction

The binary method [5] computes $Y = X^E$ using $n-1$ squarings and as many multiplications as the number of nonzero bits in the binary expansion of 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 the exponent [4, 10, 7, 6]. Recoding techniques (Booth recoding, bit-pair recoding, etc.) for sparse representations of binary numbers have been effectively used in multiplication algorithms [3, 15]. 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 [15]. 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 $\{\overline{1}, 0, 1\}$ are allowed for the digit set, in which 1 and $\overline{1}$ 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.

*Supported in part by RSA Data Security Inc., Redwood City, California.

2 Canonical Recoding

A *signed-digit* vector D of E is a sparse recoding of E using digits from the set $\{\bar{1}, 0, 1\}$. The recoding is *canonical* if D contains no adjacent nonzero digits [12, 3, 8]. Thus a canonical signed-digit vector of E is of the form $D = (D_{n-1}D_{n-2}\cdots D_0)$ with $D_i \in \{\bar{1}, 0, 1\}$ and

$$D_i \cdot D_{i-1} = 0 \text{ for } \quad 1 \le i \le n-1 \ .$$

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 [12]. 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.

Table 1: Canonical recoding.

E_{i+1}	E_i	C_i	D_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	$\bar{1}$	1
1	1	0	$\bar{1}$	1
1	1	1	0	1

As an example, when $E = 3038$, we compute the canonical signed-digit vector D as

$$\begin{aligned} E &= (0101111011110) &&= 2^{11} + 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 \ , \\ D &= (10\bar{1}0000\bar{1}000\bar{1}0) &&= 2^{12} - 2^{10} - 2^5 - 2^1 \ . \end{aligned}$$

Note that in this example the exponent E contains 9 nonzero bits while its canonically recoded version contains only 4 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. For example, the following signed-digit number for $E = 3038$ produced by the original Booth recoding algorithm contains 5 nonzero digits as opposed to 4:

$$(011000\bar{1}000\bar{1}0) \quad = \quad 2^{11} + 2^{10} - 2^6 + 2^5 - 2^1 \ .$$

Certain variations of the Booth algorithm also produce recodings which are suboptimal in terms of the number of zero digits of the recoding. For example, the first of the two algorithms given in [10] replaces the occurrences of 01^a0 by $10^{a-1}\bar{1}0$, and consequently recodes (01111011110) as $(1000\bar{1}000\bar{1}0)$. Since $(\bar{1}1) = (0\bar{1})$, the optimal recoding is $(10000\bar{1}000\bar{1}0)$. The second algorithm in [10] recodes (01111011110) correctly but is suboptimal on binary numbers in which two trails of 1s are separated by (010). For example (0111101011110) is recoded as $(1000\bar{1}011000\bar{1}0)$, which can be made more sparse by using the identity $(\bar{1}011) = (\bar{1}0\bar{1})$.

We note that Reitwiesner's canonical recoding algorithm has none of these shortcomings; the recoding D it produces by the recipe in Table 1 is provably the optimal signed-digit number [12].

3 The Standard m-ary Method

The binary method can be generalized to the (standard) m-ary method [5] 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 $E = (E_{n-1}E_{n-2}\cdots E_1E_0)$ be the binary expansion of the exponent. We will assume that the most significant bit is equal to zero, i.e., $E_{n-1} = 0$. 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

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

Note that $0 \le F_i \le 2^d - 1$ and $E = \sum_{i=0}^{k-1} F_i 2^{id}$. 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 $k \ge 1$.
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 for all $F = 2, 3, 4, \ldots, 2^d - 1$.
3. $Y := X^{F_{k-1}}$
4. **for** $i = k-2$ **downto** 0
  4a. $Y := Y^{2^d}$
  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 $(k-1)\left(\frac{m-1}{m}\right)$. Thus we find that on the average, the m-ary method requires a total of

$$T_s(n,d) \ = \ n - d + \left(\frac{n}{d} - 1\right)\left(1 - \frac{1}{2^d}\right) + 2^d - 2 \tag{2}$$

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

$$T_s(n,1) = \frac{3}{2}n - \frac{3}{2} \ , \quad T_s(n,2) = \frac{11}{8}n - \frac{3}{4} \ , \quad T_s(n,3) = \frac{31}{24}n - \frac{17}{8} \ , \tag{3}$$

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 $n = 1 + \lfloor \log_2 E \rfloor$ 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 for all possible length d bit-sections of that can appear in D.
4. $Y := X^{F_{k-1}}$
5. **for** $i = k-2$ **downto** 0
5a. $Y := Y^{2^d}$
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 $\mathcal{L}$ the formal language of all words w over the alphabet $\{\overline{1}, 0, 1\}$ in which none of the patterns

$$1\,1\ ,\ 1\,\overline{1}\ ,\ \overline{1}\,1\ ,\ \overline{1}\,\overline{1}$$

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

Lemma 1

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

Proof By considering the words in $\mathcal{L}$ according to their first letter, we see

$$\mathcal{L} = \lambda + 1 + \overline{1} + 1\,0\,\mathcal{L} + \overline{1}\,0\,\mathcal{L} + 0\,\mathcal{L} \quad , \tag{5}$$

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

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

It follows from (5) that $f_{\mathcal{L}}$ satisfies

$$f_{\mathcal{L}}(t) = 1 + 2t + 2t^2 f_{\mathcal{L}}(t) + t f_{\mathcal{L}}(t) \quad ,$$

and therefore

$$f_{\mathcal{L}}(t) = \frac{1+2t}{1-t-2t^2} \ = \ \frac{4}{3}\cdot\frac{1}{1-2t} - \frac{1}{3}\cdot\frac{1}{1+t} \ . \tag{6}$$

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

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

$$\tau_d - 3 \;=\; \frac{1}{3}\,[2^{d+2} + (-1)^{d+1}\,] - 3 \;.$$

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 preprocessing 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 m-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 8 states for the 8 possible combinations of input as given in Table 1. The state transitions given in Table 2 are produced by considering all 8 states labeled s_0 through 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.

Table 2: State transition table for the canonical recoding algorithm.

State		Output	Next State	
s_i	(E_{i+1}, E_i, C_i)	(D_i, C_{i+1})	$E_{i+2} = 0$	$E_{i+2} = 1$
s_0	$(0,0,0)$	$(0,0)$	s_0	s_4
s_1	$(0,0,1)$	$(1,0)$	s_0	s_4
s_2	$(0,1,0)$	$(1,0)$	s_0	s_4
s_3	$(0,1,1)$	$(0,1)$	s_1	s_5
s_4	$(1,0,0)$	$(0,0)$	s_2	s_6
s_5	$(1,0,1)$	$(\overline{1},1)$	s_3	s_7
s_6	$(1,1,0)$	$(\overline{1},1)$	s_3	s_7
s_7	$(1,1,1)$	$(0,1)$	s_3	s_7

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 $\mathcal{P}_{0j} = 0$ 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

$$\mathcal{P} = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} . \tag{7}$$

The limiting probability π_i of state s_i can be found by solving the system of linear equations $\pi\mathcal{P} = \pi$ with $\pi_0 + \pi_1 + \cdots + \pi_7 = 1$. This gives

$$\pi = \left[\frac{1}{6}\ ,\ \frac{1}{12}\ ,\ \frac{1}{12}\ ,\ \frac{1}{6}\ ,\ \frac{1}{6}\ ,\ \frac{1}{12}\ ,\ \frac{1}{12}\ ,\ \frac{1}{6}\right] . \tag{8}$$

Having computed π_i and $\mathcal{P}_{ij}$ for all $0 \le i, j \le 7$, 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

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

In particular, the average number of nonzero digits in the canonically recoded binary number D is equal to $\frac{n}{3}$. Therefore the average number of squarings plus multiplications required by the recoded binary method for large n is $\frac{4}{3}n + O(1)$, which is better than $T_s(n,1) = \frac{3}{2}n + O(1)$ required by the standard binary method.

Theorem 1 *The probability that a length d bit-section F in a canonically recoded signed-digit vector has all bits equal to zero is*

$$\left(\frac{1}{2}\right)^{d-1}\ \frac{2}{3}\ =\ \frac{1}{3 \cdot 2^{d-2}} .$$

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

$$\mathcal{P}(D_{i+1} = 0 \mid D_i = 0)\ =\ \frac{\displaystyle\sum_{j=0,3,4,7} \pi_0\mathcal{P}_{0j} + \pi_3\mathcal{P}_{3j} + \pi_4\mathcal{P}_{4j} + \pi_7\mathcal{P}_{7j}}{\pi_0 + \pi_3 + \pi_4 + \pi_7} = \frac{1}{2} .$$

It follows that for $d \ge 1$

$$\mathcal{P}(D_{i+d-1} = 0 \mid D_{i+d-2} = 0, D_{i+d-3} = 0, \ldots, D_i = 0)\ =\ \left(\frac{1}{2}\right)^{d-1}\ \frac{2}{3} .$$

□

Combining Theorem 1 and the preprocessing cost for the recoded m-ary method given in Lemma 2, we find that on the average the recoded m-ary method with $m = 2^d$ requires a total of

$$T_r(n,d) = n - d + \left(1 - \frac{1}{3 \cdot 2^{d-2}}\right)\left(\frac{n}{d} - 1\right) + \frac{1}{3}\left[2^{d+2} + (-1)^{d+1}\right] - 3 \tag{9}$$

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 $n = 2^8, 2^9, \ldots, 2^{16}$ 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

$$T_r(n,1) = \frac{4}{3}\,n - \frac{4}{3}\ , \qquad T_r(n,2) = \frac{4}{3}\,n - \frac{2}{3}\ , \qquad T_r(n,3) = \frac{23}{18}\,n + \frac{75}{18}\ , \tag{10}$$

respectively.

6 Comparison of Standard and Recoded m-ary Methods

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

Table 3: The average number of multiplications for the recoded and standard m-ary methods.

$d = \log_2 m$	1	2	3	4	5	6	7	8
$T_s(n,d)/n$	1.5	1.375	1.29167	1.23437	1.19375	1.16406	1.14174	1.12451
$T_r(n,d)/n$	1.33333	1.33333	1.27778	1.22917	1.19167	1.16319	1.14137	1.12435

We can compute directly from the expressions in (2) and (9) that for constant d

$$\lim_{n\to\infty} \frac{T_r(n,d)}{T_s(n,d)} \ = \ \frac{(d+1)2^d - \frac{4}{3}}{(d+1)2^d - 1} \ < \ 1\ . \tag{11}$$

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

$$\frac{T_r(n,d_r)}{T_s(n,d_s)} \ > \ 1 \tag{12}$$

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

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

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

$$d_s^2\, 2^{d_s}\, \log 2 \approx n \qquad \text{and} \qquad d_r^2\, 2^{d_r}\, \log 2 \approx \frac{3}{4} n\ . \tag{13}$$

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

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

for large n, which implies (12). Exact values of d_s and d_r for a given n can be obtained by enumeration. These optimal values of d_s and d_r are given in Table 4 together with the corresponding values of T_s and T_r for each $n = 2^8, 2^9, \ldots, 2^{16}$.

Table 4: Optimal values of d_s and d_r together with T_s and T_r.

n	d_s	$T_s(n, d_s)$	d_r	$T_r(n, d_r)$
128	4	168	3	168
256	4	326	4	328
512	5	636	4	643
1024	5	1247	5	1255
2048	6	2440	6	2458
4096	7	4795	7	4836
8192	8	9457	7	9511
16384	8	18669	8	18751
32768	9	36902	9	37070
65536	10	73095	10	73433

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 [13], 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 extended Euclid algorithm, the cost of this computation far exceeds the time gained by the use of the recoding technique in exponentiation. Thus, at this time the recoding techniques do not seem to be particularly applicable to these cryptosystems. However, the recoding techniques may be useful for computations on elliptic curves over finite fields since in these cases the inverse is available at no additional cost [10, 6]. 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

[1] A. D. Booth. A signed binary multiplication technique. *Q. J. Mech. Appl. Math.*, 4(2):236–240, 1951. (Also reprinted in [14], pp. 100–104).

[2] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.

[3] K. Hwang. *Computer Arithmetic, Principles, Architecture, and Design*. New York, NY: John Wiley & Sons, 1979.

[4] J. Jedwab and C. J. Mitchell. Minimum weight modified signed-digit representations and fast exponentiation. *Electronics Letters*, 25(17):1171–1172, 17th August 1989.

[5] D. E. Knuth. *The Art of Computer Programming,* Volume 2: *Seminumerical Algorithms*. Reading, MA: Addison-Wesley, Second edition, 1981.

[6] N. Koblitz. CM-curves with good cryptographic properties. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO 91, Proceedings*, Lecture Notes in Computer Science, No. 576, pages 279–287. New York, NY: Springer-Verlag, 1991.

[7] Ç. K. Koç. High-radix and bit recoding techniques for modular exponentiation. *International Journal of Computer Mathematics*, 40(3+4):139–156, 1991.

[8] I. Koren. *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[9] O. L. MacSorley. High-speed arithmetic in binary computers. *Proceedings of the IRE*, 49:67–91, January 1961. (Also reprinted in [14], pp. 14–38).

[10] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. Rapport de Recherche 983, INRIA, March 1989.

[11] National Institute for Standards and Technology. Digital signature standard (DSS). *Federal Register*, 56:169, August 1991.

[12] G. W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[13] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[14] E. E. Swartzlander, editor. *Computer Arithmetic*, volume I. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[15] S. Waser and M. J. Flynn. *Introduction to Arithmetic for Digital System Designers*. New York, NY: W. H. Freeman and Company, 1982.
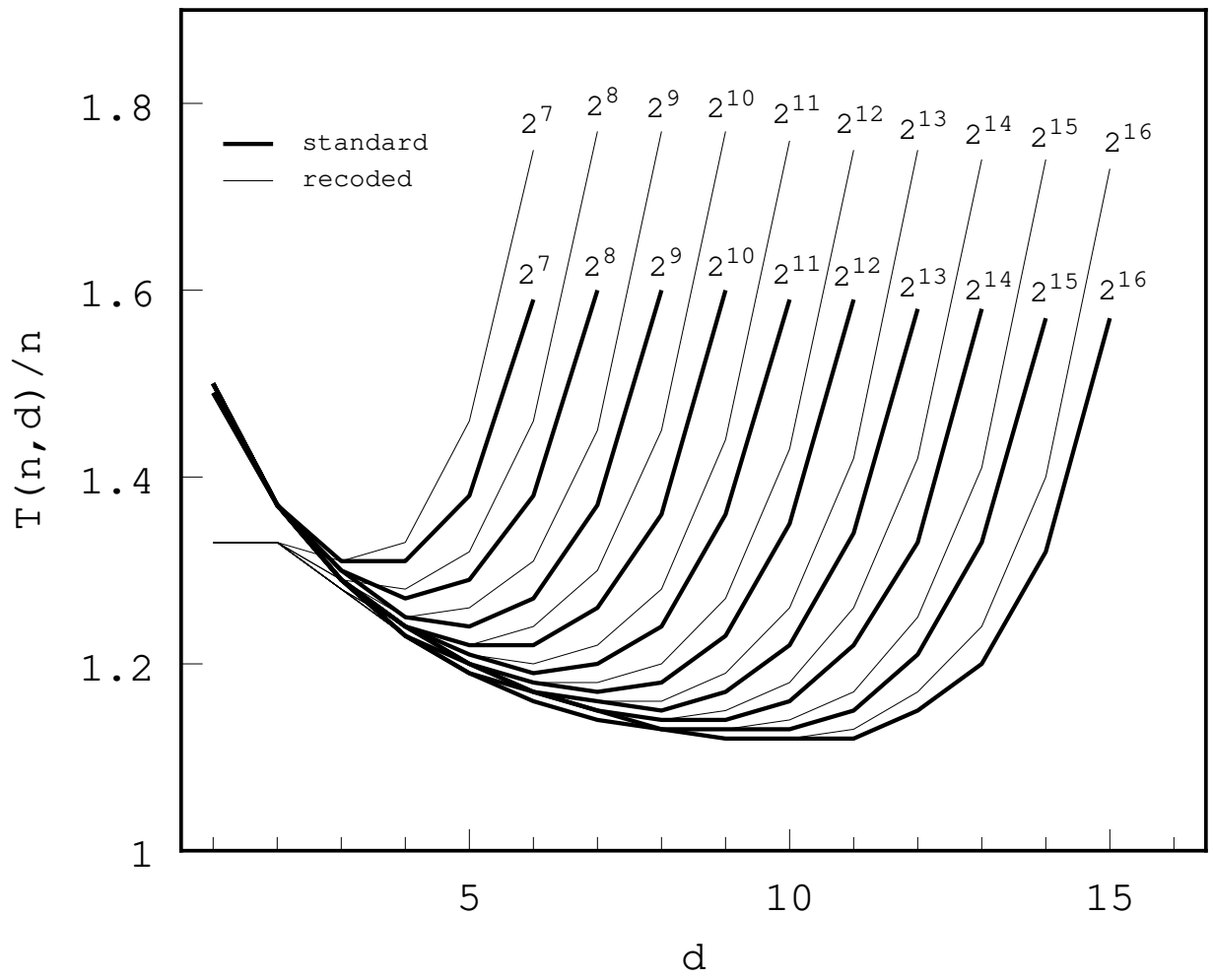
Figure 1: The standard versus recoded m-ary methods.