New Minimal Modified Radix-r Representation with Applications to Smart Cards

Marc Joye[1] and Sung-Ming Yen[2]

[1] Gemplus Card International, Card Security Group
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos Cedex, France
marc.joye@gemplus.com
http://www.gemplus.com/smart/
http://www.geocities.com/MarcJoye/
[2] Laboratory of Cryptography and Information Security (LCIS)
Dept of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320, R.O.C.
yensm@csie.ncu.edu.tw
http://www.csie.ncu.edu.tw/~yensm/

Abstract. This paper considers the problem of finding a minimum-weighted representation of an integer under any modified radix-r number system. Contrary to existing methods, the proposed transformation is carried out from the left to the right (i.e., from the most significant position). This feature finds numerous applications and especially in fast arithmetic techniques because it reduces both time and space complexities, which is particularly attractive for small devices like smart cards.

1 Introduction

A *modified radix-r representation* (MR-r) of an integer N is a sequence of digits $N = (\ldots, c_2, c_1, c_0)$ with $-r < c_i < r$. Unlike the (usual) radix-r representation (i.e., with $0 \le c_i < r$), such a representation is not unique. The radix-r representation is a special case of MR-r representation.

In the theory of arithmetic codes [vL82] or for fast implementation of cryptosystems [Gor98,MvOV97], it is of interest to have a representation such that the number of nonzero digits (i.e., the *arithmetic weight* [CL73]) is minimal. In the binary case (i.e., when $r = 2$), a well-known minimal representation is given by the so-called *nonadjacent form* (NAF), that is, a representation with $c_i \cdot c_{i+1} = 0$ for all $i \ge 0$. In [Rei60], Reitwiesner proved that each integer has exactly one NAF. Clark and Liang [CL73] later addressed the general case and extended the notion of NAF to an arbitrary radix $r \ge 2$:

Definition 1. *A* MR-r *representation* $(\ldots, c_2, c_1, c_0)$ *is said to be a* generalized nonadjacent form *(*GNAF*) if and only if*

(G1) $|c_i + c_{i+1}| < r$ *for all* i *; and*
(G2) $|c_i| < |c_{i+1}|$ *if* $c_i \cdot c_{i+1} < 0$.

As one can easily see, this form coincides with the definition of the NAF when $r = 2$. Moreover, as for the NAF, it can be proven that this form is unique and has minimal arithmetic weight [CL73].

However, the GNAF is not the only representation with minimal arithmetic weight. For example, $(1, 0, 1, 0, -1, 0)$ and $(1, 0, 0, 1, 1, 0)$ are two minimal MR-2 representations for 38.

Our Results

This paper considers a new *minimal* MR-r representation and presents an efficient algorithm to compute it. This new representation, unlike the GNAF, presents the nice property to be obtained by scanning the digits *from the left to the right* (i.e., from the most significant digit to the least significant one). This processing direction is of great importance since only for that direction a table of precomputed values may be used to speed up the exponentiation, at least for exponentiation algorithms processing one digit at a time [BGMW92]. A subsequent advantage is that the exponent need not be recoded in advance, which results in better performances in both running time and memory space. This is especially important for small devices like the smart cards. Moreover, only for that direction, further speedups may be obtained if the element to be raised up presents a special structure [Coh93, pp. 9–10]. Finally, this processing direction also solves an open problem introduced in [WH97, § 3.6].

2 New Minimal Representation

Throughout this paper, for convenience, $-a$ will sometimes be denoted as $\bar{a}$ and $\langle S\rangle^k$ will represent $S, S, \ldots, S$ (k times).

2.1 Elementary Blocks

Given the radix-r representation $(\ldots, c_2, c_1, c_0)$ of an integer N, the corresponding GNAF can easily be obtained as follows [CL73]: compute the radix-r representation of $(r + 1)N$, $(\ldots, b_2, b_1, b_0)$, then the GNAF is given by $(\ldots, c'_2, c'_1, c'_0)$ where $c'_i = b_{i+1} - c_{i+1}$. So, if the radix-r representation of $(r + 1)N$ is known, we are done. This computation can be carried out by right-to-left adding $N = (\ldots, c_2, c_1, c_0)$ and $rN = (\ldots, c_2, c_1, c_0, 0)$ according to the standard carry rule [Knu81, p. 251]:

$$\begin{cases} \kappa_0 = 0 \\ \kappa_{i+1} = \left\lfloor \dfrac{c_i + c_{i+1} + \kappa_i}{r} \right\rfloor \end{cases}, \tag{1}$$

$b_0 = c_0$ and $b_{i+1} = c_i + c_{i+1} + \kappa_i - r\kappa_{i+1}$ for $i \geq 0$. Left-to-right algorithms to add integers also exist [Knu81, Exercises 4.3.1.5 and 4.3.1.6]. However, they do not consistently output one digit per iteration; delay may occur when the sum of two adjacent digits is equal to $r - 1$. For that reason, an *on-line* (i.e., digit-by-digit) left-to-right computation of the GNAF seems to be impossible.

This paper considers a quite different approach: instead of trying to obtain the GNAF, we are looking for other minimal forms that may be efficiently evaluated from the left to the right. Our technique relies on the following observation:

Proposition 1. *Let $(\ldots, c_2, c_1, c_0)$ be the radix-r representation of an integer N, and let $(c_{f+1}, c_f, \ldots, c_{e+1}, c_e)$ be a subchain of digits of this representation such that*

(E1) $f > e$;
(E2) $c_e + c_{e+1} \neq r - 1$;
(E3) $c_j + c_{j+1} = r - 1$ *for* $e < j < f$;
(E4) $c_f + c_{f+1} \neq r - 1$.

Then all but the first and the last digits of the corresponding GNAF *are entirely determined.*

Proof. We note from Eq. (1) that, since $\kappa_i = 0$ or 1, the value of the carry-out κ_{i+1} does not depend on the carry-in κ_i for $i \geq e$. Indeed, we have $\kappa_{e+1} = \lfloor \frac{c_e + c_{e+1}}{r} \rfloor$ by Condition (E2); hence $\kappa_{j+1} = \kappa_{e+1}$ for $e \leq j < f$ by induction from Condition (E3); and $\kappa_{f+1} = \lfloor \frac{c_f + c_{f+1}}{r} \rfloor$ by Condition (E4). Therefore, if $(\star, c'_f, \ldots, c'_{e+1}, \star)$ denotes the digits corresponding to the GNAF, it follows that

$$\begin{aligned} c'_j &= b_{j+1} - c_{j+1} = c_j + \kappa_j - r\kappa_{j+1} \\ &= \begin{cases} c_j + (1-r)\kappa_{e+1} & \text{for } e+1 \leq j \leq f-1 \\ c_f + \kappa_{e+1} - r\kappa_{f+1} & \text{for } j = f \end{cases} \end{aligned} \tag{2}$$

where $\kappa_{e+1} = \lfloor \frac{c_e + c_{e+1}}{r} \rfloor$ and $\kappa_{f+1} = \lfloor \frac{c_f + c_{f+1}}{r} \rfloor$. □

A subchain of radix-r digits satisfying Conditions (E1)–(E4) will be called a *radix-r elementary block*. From Proposition 1, two types of elementary blocks may be distinguished.

Definition 2. *Let $0 \leq d, b, e \leq r - 1$, $c = r - 1 - d$ (that is, $c + d = r - 1$) and $k \geq 0$. An elementary block of the form*

$$(b, d, \langle c, d \rangle^k, e) \quad \textit{with } b + d \neq r - 1 \textit{ and } e + d \neq r - 1 \tag{3}$$

will be referred to as a Type 1 radix-r elementary block*; and an elementary block of the form*

$$(b, d, \langle c, d \rangle^k, c, e) \quad \textit{with } b + d \neq r - 1 \textit{ and } e + c \neq r - 1 \tag{4}$$

as a Type 2 radix-r elementary block.

Notice that a Type 1 elementary block contains an odd number of digits (and at least 3) while a Type 2 elementary block contains an even number of digits (and at least 4).

Based on this definition, we can now present the new recoding algorithm.

2.2 General Recoding Algorithm

From Proposition 1 (see also Eq. (2)), the GNAF corresponding to a Type 1 elementary block $(b, d, \langle c, d\rangle^k, e)$ is given by[1]

$$G_1 = (\star, d + \lfloor \tfrac{e+d}{r} \rfloor - r\lfloor \tfrac{b+d}{r} \rfloor, \left\langle c + (1-r)\lfloor \tfrac{e+d}{r} \rfloor, d + (1-r)\lfloor \tfrac{e+d}{r} \rfloor \right\rangle^k, \star) \ . \tag{5}$$

By definition of a Type 1 elementary block, we have $b + d \neq r - 1$ and $e + d \neq r - 1$. Hence, Eq. (5) respectively simplifies to

$$G_1 = \begin{cases} (\star, d, \langle c, d\rangle^k, \star) & \text{if } b+d < r-1 \text{ and } e+d < r-1 \\ (\star, d+1, \langle -d, -c\rangle^k, \star) & \text{if } b+d < r-1 \text{ and } e+d \geq r \\ (\star, d-r, \langle c, d\rangle^k, \star) & \text{if } b+d \geq r \text{ and } e+d < r-1 \\ (\star, -c, \langle -d, -c\rangle^k, \star) & \text{if } b+d \geq r \text{ and } e+d \geq r \end{cases} . \tag{6}$$

Similarly, the GNAF corresponding to a Type 2 block $(b, d, \langle c, d\rangle^k, c, e)$ is given by

$$G_2 = \begin{cases} (\star, d, \langle c, d\rangle^k, c, \star) & \text{if } b+d < r-1 \text{ and } e+c < r-1 \\ (\star, d+1, \langle -d, -c\rangle^k, -d, \star) & \text{if } b+d < r-1 \text{ and } e+c \geq r \\ (\star, d-r, \langle c, d\rangle^k, c, \star) & \text{if } b+d \geq r \text{ and } e+c < r-1 \\ (\star, -c, \langle -d, -c\rangle^k, -d, \star) & \text{if } b+d \geq r \text{ and } e+c \geq r \end{cases} . \tag{7}$$

Here too, we see the difficulty of converting a radix-r representation into its GNAF by scanning the digits from the left to the right. If an elementary block begins with $(b, d, c, d, c, d, \ldots)$ with $b + d < r - 1$, the output will be $(\star, d, c, d, c, d, \ldots)$ or $(\star, d{+}1, -d, -c, -d, -c, \ldots)$ depending on the two last digits of the elementary block; if $b + d \geq r$, the output will be $(\star, d - r, c, d, c, d, \ldots)$ or $(\star, -c, -d, -c, -d, -c, \ldots)$. However, as stated in the next lemma, these outputs may be replaced by *equivalent* forms (i.e., forms having the same length and the same weight, and representing the same integer) so that the two last digits of an elementary block do not need to be known in advance.

Lemma 1. *Let* $0 \leq d \leq r - 1$ *and* $c = r - 1 - d$. *Then*

(i) if $d \neq r - 1$, $(d + 1, \langle -d, -c\rangle^k)$ *and* $(\langle d, c\rangle^k, d + 1)$ *are equivalent* MR-*r* *representations ;*

(ii) if $d \neq 0$, $(d - r, \langle c, d\rangle^k)$ *and* $(\langle -c, -d\rangle^k, d - r)$ *are equivalent* MR-*r* *representations.*

Proof. With the MR-*r* notation, $(d + 1, \langle -d, -c\rangle^k)$ represents the integer

$$\begin{aligned} N &= (d+1)r^{2k} + \textstyle\sum_{j=0}^{k-1}(-dr - c)r^{2j} \\ &= (d+1)r^{2k} + \textstyle\sum_{j=0}^{k-1}(cr + d + 1 - r^2)r^{2j} \\ &= d\,r^{2k} + \textstyle\sum_{j=0}^{k-1}(cr + d)r^{2j} + 1 \quad \text{since } \textstyle\sum_{j=0}^{k-1} r^{2j} = \frac{r^{2k}-1}{r^2-1} \end{aligned}$$

which is thus also represented by $(d, \langle c, d\rangle^{k-1}, c, d + 1)$. We also note that $(d + 1, \langle -d, -c\rangle^k)$ and $(\langle d, c\rangle^k, d{+}1)$ have the same arithmetic weight since their digits are identical, in absolute value. The second equivalence is proved similarly. □

[1] The '$\star$' indicates that the digit is unknown.

Consequently, GNAFs G_1 and G_2 (see Eqs (6) and (7)) can respectively be replaced by another equivalent form, which we call *generalized star form* (GSF), respectively given by

$$S_1 = \begin{cases} (\star, \langle d, c\rangle^k, d, \star) & \text{if } b+d < r-1 \text{ and } e+d < r-1 \\ (\star, \langle d, c\rangle^k, d+1, \star) & \text{if } b+d < r-1 \text{ and } e+d \geq r \\ (\star, \langle -c, -d\rangle^k, d-r, \star) & \text{if } b+d \geq r \text{ and } e+d < r-1 \\ (\star, \langle -c, -d\rangle^k, -c, \star) & \text{if } b+d \geq r \text{ and } e+d \geq r \end{cases} \tag{8}$$

and

$$S_2 = \begin{cases} (\star, \langle d, c\rangle^k, d, c, \star) & \text{if } b+d < r-1 \text{ and } e+c < r-1 \\ (\star, \langle d, c\rangle^k, d+1, -d, \star) & \text{if } b+d < r-1 \text{ and } e+c \geq r \\ (\star, \langle -c, -d\rangle^k, d-r, c, \star) & \text{if } b+d \geq r \text{ and } e+c < r-1 \\ (\star, \langle -c, -d\rangle^k, -c, -d, \star) & \text{if } b+d \geq r \text{ and } e+c \geq r \end{cases} . \tag{9}$$

The outputs now behave very regularly; an elementary block $(b, d, c, d, c, d, \ldots)$ will be recoded into $(\star, d, c, d, c, d, \ldots)$ if $b + d < r - 1$ or into $(\star, -c, -d, -c, -d, -c, \ldots)$ if $b + d \geq r$. We have just to take some precautions when outputting the last digits of the corresponding GSFs. The following example clarifies the technique.

Example 1. Suppose that the radix-4 GSF of $N = 208063846$ has to be computed. Its radix-4 representation is $(30121230311212)_4$. Then, by adding artificial beginning and ending 0's and decomposing this representation into elementary blocks, the corresponding GSF is easily obtained from Eqs (8) and (9) as follows.

$$\begin{array}{lccccccccccccccccc}
\text{Radix-4 representation:} & 0&0&3&0&1&2&1&2&3&0&3&1&1&2&1&2&.0 \\
\text{Elementary blocks:} & 0&0&3&0&1&&&&&&&&&&&& \\
 & &&&0&1&2&1&2&3&&&&&&&& \\
 & &&&&&&&2&3&0&3&1&&&&& \\
 & &&&&&&&&&&3&1&1&&&& \\
 & &&&&&&&&&&&1&1&2&1&2&.0 \\
\text{Corresponding GSF blocks:} & \star&0&3&0&\star&&&&&&&&&&&& \\
 & &&&\star&1&2&2&\bar{1}&\star&&&&&&&& \\
 & &&&&&&&\star&0&\bar{3}&0&\star&&&&& \\
 & &&&&&&&&&&\star&\bar{3}&\star&&&& \\
 & &&&&&&&&&&&\star&1&2&1&2&\star \\
\hline
\text{Radix-4 GSF representation:} & &&3&0&1&2&2&\bar{1}&0&\bar{3}&0&\bar{3}&1&2&1&2&
\end{array}$$

□

In the previous example, it clearly appears that, from the definition of an elementary block, the two first digits of an elementary block are the two last ones of the previous block. This also illustrates that the decomposition into elementary (and thus GSF) blocks always exists. Note also that the values of the first and last digits of the corresponding GSF are given by the adjacent GSF blocks.

So, by carefully considering the two last digits of each possible elementary block, we finally obtain the desired algorithm. Two additional variables are used: variable β_i plays a role similar to the "borrow" and τ keeps track the value of the repeated digit (d in Definition 2).

```
INPUT:  $(n_{m-1}, \dots, n_0)$          (Radix-$r$ representation)
OUTPUT: $(n'_m, n'_{m-1}, \dots, n'_0)$  (Radix-$r$ GSF representation)

  $\beta_m \leftarrow 0$; $n_m \leftarrow 0$; $n_{-1} \leftarrow 0$; $n_{-2} \leftarrow 0$, $\tau \leftarrow 0$
  for $i$ from $m$ down to $0$ do
    case
      $n_i + n_{i-1} < r - 1$: $\beta_{i-1} \leftarrow 0$, $\tau \leftarrow n_{i-1}$
      $n_i + n_{i-1} = (r-1)$: $\beta_{i-1} \leftarrow \begin{cases} 0 & \text{if } \beta_i = 1,\ n_{i-1} = (r-1-\tau) \\ & \quad \text{and } n_{i-1} + n_{i-2} < r-1 \\ 1 & \text{if } \beta_i = 0,\ n_{i-1} = (r-1-\tau) \\ & \quad \text{and } n_{i-1} + n_{i-2} \geq r \\ \beta_i & \text{otherwise} \end{cases}$
      $n_i + n_{i-1} \geq r$: $\beta_{i-1} \leftarrow 1$, $\tau \leftarrow n_{i-1}$
    endcase
    $n'_i \leftarrow -r\beta_i + n_i + \beta_{i-1}$
  od
```

Fig. 1. Left-to-right radix-r GSF recoding algorithm.

To verify the correctness of the algorithm, it suffices to check that each type of elementary block is effectively transformed accordingly to Eqs (8) and (9). The next theorem states that the proposed algorithm is optimal in the sense that the resulting representation has the fewest number of nonzero digits of any MR-r representation.

Theorem 1. *The* GSF *of a number has minimal arithmetic weight.*

Proof. This is straightforward by noting the GSF is obtained from the GNAF where some subchains were replaced according to Lemma 1. Since these transformations produce equivalent subchains, the GSF has the same arithmetic weight as the GNAF and is thus minimal. □

Remark 1. We note that, as in [CL73], the proposed algorithm can be extended to transform an arbitrary MR-r representation into a minimal one.

2.3 Binary Case

In the binary case (i.e., when $r = 2$), the algorithm presented in Fig. 1 is slightly simpler. Using the same notations as in Fig. 1, if $n_i + n_{i-1} = 1$, we can easily verify that

$$\beta_{i-1} = \left\lfloor \frac{\beta_i + n_{i-1} + n_{i-2}}{2} \right\rfloor \tag{10}$$

is a valid expression for β_{i-1}. Moreover, this expression remains valid when $n_i = n_{i-1} = 0$ (i.e., in the case $n_i + n_{i-1} < 1$) because if $n_{i+1} = 0$ then $\beta_i = 0$ and if $n_{i+1} = 1$ then, by Eq. (10), we also have $\beta_i = 0$; therefore Eq. (10) yields $\beta_{i-1} = 0$. Similarly when $n_i = n_{i-1} = 1$ (i.e., $n_i + n_{i-1} \geq 2$), we can show that Eq. (10) yields $\beta_{i-1} = 1$, as expected. The radix-2 GSF recoding algorithm thus becomes [JY00]:

```
INPUT:  $(n_{m-1}, \ldots, n_0)$       (Binary representation)
OUTPUT: $(n'_m, n'_{m-1}, \ldots, n'_0)$ (Binary GSF representation)

  $\beta_m \leftarrow 0$; $n_m \leftarrow 0$; $n_{-1} \leftarrow 0$; $n_{-2} \leftarrow 0$
  for $i$ from $m$ down to 0 do
    $\beta_{i-1} \leftarrow \lfloor \frac{\beta_i + n_{i-1} + n_{i-2}}{2} \rfloor$
    $n'_i \leftarrow -r\beta_i + n_i + \beta_{i-1}$
  od
```

Fig. 2. Left-to-right binary GSF recoding algorithm.

3 Applications

Minimal representations naturally find applications in the theory of arithmetic codes [vL82] and in fast arithmetic techniques [Boo51,Avi61]. In this section, we will restrict our attention to fast exponentiation (see the excellent survey article of [Gor98]).

The commonly used methods to compute g^k are the generalized "square-and-multiply" algorithms. When base g is fixed or when the computation of an inverse is virtually free, as for elliptic curves [MO90], a signed-digit representation further speeds up the computation. These algorithms input the MR-r representations of exponent $k = (k_m, \ldots, k_0)$ (i.e., with $-r < k_i < r$ and $k_m \neq 0$) and of base g, and output $y = g^k$. This can be carried out from right to left (Fig. 3a) or from left to right (Fig. 3b).

```
INPUT:  $k = (k_m, \ldots, k_0), g$
OUTPUT: $y = g^k$

  $y \leftarrow 1$; $h \leftarrow g$
  for $i$ from 0 to $m - 1$ do
    $y \leftarrow y \cdot h^{k_i}$
    $h \leftarrow h^r$
  od
  $y \leftarrow y \cdot h^{k_m}$
```

(a) Right-to-left (RL).

```
INPUT:  $k = (k_m, \ldots, k_0), g$
OUTPUT: $y = g^k$

  $y \leftarrow g^{k_m}$
  for $i$ from $m - 1$ down to 0 do
    $y \leftarrow y^r$
    $y \leftarrow y \cdot g^{k_i}$
  od
```

(b) Left-to-right (LR).

Fig. 3. Modified exponentiation algorithms.

Using Markov chains, Arno and Wheeler [AW93] precisely estimated that the average proportion of nonzero digits in the radix-r GNAF (and thus also in the radix-r GSF) representation is equal to

$$\rho_r = \frac{r-1}{r+1} \ . \tag{11}$$

The LR algorithm can use the precomputation of some g^t for $-r < t < r$ while the RL algorithm cannot! One could argue that the exponent may first be recoded into a minimal MR-r representation and then the LR algorithm be applied. However this also requires more memory space since each digit in the MR-r requires an additional bit (to encode the sign for each digit). Consequently, since only the LR algorithm is efficient, the proposed (left-to-right) general recoding algorithm (Fig. 1) will also bring some advantages because, as aforementioned, the exponent need not be pre-recoded into its MR-r representation. This feature is especially desirable for small devices, like smart cards.

Another generalization of exponentiation algorithms is to use variable windows to skip strings of consecutive zeros. Here too, the proposed recoding algorithms offer some speedups. Indeed, as a side effect, while having the same weight as the GNAF, the GSF representation has more adjacent nonzero digits, which increases the length of the runs of zeros. This property was successfully applied by Koyama and Tsuruoka [KT93] to speed up the window exponentiation on elliptic curves. We note nevertheless that the zero runs of their algorithm are generally longer than those obtained by the GSFs.

4 Conclusions

In this paper, a new modified representation of integers for a general radix $r \geq 2$ is developed. Like the nonadjacent form (NAF) and its generalization, the proposed representation has the fewest number of nonzero digits of any modified representation.

When developing fast computational algorithms for hardware implementation, especially for some small devices which have very limited amount of memory, the problem of extra space requirement cannot be overlooked. Scanning and transforming the original representation from the most significant digit has its merit in not storing the resulting minimum-weighted result, hence memory space requirements are reduced. Finally, this solves a problem considered to be hard (see [WH97, § 3.6]), i.e., to obtain a minimal modified radix-4 representation by scanning the digits of the standard representation from the left to the right.

Acknowledgments

The authors are grateful to the anonymous referees for their useful comments. Sung-Ming Yen was supported in part by the National Science Council of the Republic of China under contract NSC 87-2213-E-032-013.

References

Avi61. A. Avizienis, *Signed-digit number representations for fast parallel arithmetic*, IRE Transactions on Electronic Computers **EC-10** (1961), 389–400, Reprinted in [Swa90, vol. II, pp. 54–65].

AW93. S. Arno and F.S. Wheeler, *Signed digit representations of minimal Hamming weight*, IEEE Transactions on Computers **C-42** (1993), no. 8, 1007–1010.

BGMW92. E.F. Brickell, D.M. Gordon, K.S. McCurley, and D.B. Wilson, *Fast exponentiation with precomputation*, Advances in Cryptology – EUROCRYPT '92, Lecture Notes in Computer Science, vol. 658, Springer-Verlag, 1992, pp. 200–207.

Boo51. A.D. Booth, *A signed binary multiplication technique*, The Quaterly Journal of Mechanics and Applied Mathematics **4** (1951), 236–240, Reprinted in [Swa90, vol. I, pp. 100–104].

CL73. W.E. Clark and J.J. Liang, *On arithmetic weight for a general radix representation of integers*, IEEE Transactions on Information Theory **IT-19** (1973), 823–826.

Coh93. H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, 1993.

EK94. Ö. Eğecioğlu and Ç.K. Koç, *Exponentiation using canonical recoding*, Theoretical Computer Science **129** (1994), no. 2, 407–417.

GHM96. D. Gollmann, Y. Han, and C.J. Mitchell, *Redundant integer representation and fast exponentiation*, Designs, Codes and Cryptography **7** (1996), 135–151.

Gor98. D.M. Gordon, *A survey of fast exponentiation methods*, Journal of Algorithms **27** (1998), 129–146.

JY00. M. Joye and S.-M. Yen, *Optimal left-to-right binary signed-digit recoding*, IEEE Transactions on Computers **49** (2000), no. 7, 740–748.

Knu81. D.E. Knuth, *The art of computer programming/Seminumerical algorithms*, 2nd ed., vol. 2, Addison-Wesley, 1981.

KT93. K. Koyama and Y. Tsurukoa, *Speeding up elliptic cryptosystems by using a signed binary window method*, Advances in Cryptology – CRYPTO '92, Lecture Notes in Computer Science, vol. 740, Springer-Verlag, 1993, pp. 345–357.

MO90. F. Morain and J. Olivos, *Speeding up the computations on an elliptic curve using addition-subtraction chains*, Theoretical Informatics and Applications **24** (1990), 531–543.

MvOV97. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of applied cryptography, ch. 14, CRC Press, 1997.

Rei60. G.W. Reitwiesner, *Binary arithmetic*, Advances in Computers **1** (1960), 231–308.

Swa90. E.E. Swartzlander, Jr. (ed.), *Computer arithmetic*, vol. I and II, IEEE Computer Society Press, 1990.

vL82. J.H. van Lint, *Introduction to coding theory*, Springer-Verlag, 1982.

WH97. H. Wu and M.A. Hasan, *Efficient exponentiation of a primitive root in* GF(2^m), IEEE Transactions on Computers **C-46** (1997), no. 2, 162–172.