Appears in C. Park, S. Chee (Eds.): *Information Security and Cryptology – ICISC 2004,* Springer-Verlag LNCS 3506, pp. 137–153, ISBN-13 978-3-540-26226-8, 2005.

Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation

Bodo Möller*

University of California, Berkeley
bmoeller@acm.org

Abstract. This paper extends results concerning efficient exponentiation in groups where inversion is easy (e.g. in elliptic curve cryptography). It examines the right-to-left and left-to-right signed fractional window (RL-SFW and LR-SFW) techniques and shows that both RL-SFW and LR-SFW representations have minimal weight among all signed-digit representations with digit set $\{\pm 1, \pm 3, \ldots, \pm m, 0\}$. (Fractional windows generalize earlier sliding-window techniques, providing more flexibility for exponentiation algorithms in order to make best use of the memory that is available for storing intermediate results.) Then it considers the length of representations: LR-SFW representations are an improvement over RL-SFW representations in that they tend to be shorter; further length improvements are possible by post-processing the representations.

Keywords: Efficient implementations, elliptic curve cryptography.

1 Introduction

Many public-key cryptosystems involve *exponentiation* in some finite group, and often (e.g. for elliptic curve cryptography) group elements are represented such that the inversion operation is almost immediate. It is well known that *signed-digit representations* of integers e are useful to perform exponentiations g^e in such groups. A particular signed-digit representation is the right-to-left signed fractional window (RL-SFW) representation introduced in [14]. Fractional windows provide more flexibility for exponentiation algorithms than the previously known sliding-window representations; the purpose of this flexibility is to make best use of the memory that is available for storing intermediate results. The present paper also considers the left-to-right signed fractional window (LR-SFW) representation (cf. [19] and [9]); it complements [14] by proving minimality of weight for both RL-SFW and LR-SFW representations. A general motive for preferring the left-to-right variant is that it generates the digits in the order in which they are usually needed for exponentiation. We also examine the length of

* Supported by a DAAD (German Academic Exchange Service) Postdoc fellowship.

representations and see that the LR-SFW method actually provides an improvement in this respect, and that further improvements are possible. Finally, we observe that no finite-state machine can always generate a minimal-length representation among the representations of minimal weight employing a prescribed set of signed digits.

To motivate and explain the goals of this paper, let us first look at typical exponentiation techniques in some detail. Given integers $b_\ell, \ldots, b_0$ as *digits,* we write $(b_\ell \ldots b_0)_2$ for $\sum_{0 \le i \le \ell} b_i \cdot 2^i$. For $m \ge 1$ odd, let

$$B_m = \{\pm 1, \pm 3, \ldots, \pm m\}$$

be the set of odd integers with absolute values up to m. We call $b_\ell \ldots b_0$ a B_m*-representation* of e if $b_\ell, \ldots, b_0 \in B_m \cup \{0\}$ and $e = (b_\ell \ldots b_0)_2$. For example, for any m, $100\bar{1}$ is a B_m-representation of 7, where we use the convention that $\bar{b}$ denotes digit value $-b$. If we assume that $e \ne 0$ and that ℓ is chosen minimal (so that $b_\ell \ne 0$), the power g^e can be computed by the two-stage algorithm shown in Fig. 1. This algorithm processes the digits b_i from the most significant down to

```
{ LR-exponentiation: compute g^e where e = (b_ℓ ... b_0)_2, b_ℓ ≠ 0 }

{ Precomputation stage }
A ← g^2
G_1 ← g
for b = 3 to m step 2 do
    G_b ← G_{b-2} · A    { = g^b }

{ Evaluation stage }
A ← (b_ℓ > 0 ? G_{b_ℓ} : G_{|b_ℓ|}^{-1})
for i = ℓ − 1 down to 0 do
    A ← A^2
    if b_i ≠ 0 then
        A ← A · (b_i > 0 ? G_{b_i} : G_{|b_i|}^{-1})
return A
```

Fig. 1. LR-exponentiation

the least significant one, i.e. *left to right* assuming big-endian notation; we speak of *LR-exponentiation.* Let $\mathcal{H}(b_\ell \ldots b_0)$ denote the *weight* (generalized Hamming weight) of the given B_m-representation, i.e. the number of non-zero digits. The LR-exponentiation algorithm in Fig. 1 performs the following numbers of group operations (where we distinguish between squarings and general multiplications since they will typically have different computational cost, but neglect inversions as these are assumed to be almost immediate):

– In the *precomputation stage,* one squaring and $\dfrac{m-1}{2}$ general multiplications;

– in the *evaluation stage,* ℓ squarings and $\mathcal{H}(b_\ell \dots b_0) - 1$ general multiplications.[1]

Increasing parameter m makes additional digit values available, typically allowing for lower-weight representations at the cost of an increased precomputation stage effort. Parameter m also determines the amount of memory needed for storing the precomputed values $G_1, \dots, G_m$, so implementations may have to take into account some upper limit on m.

For given m and e, there is a lower limit on ℓ (i.e. there is a lower limit on the *length* $\ell + 1$ of B_m-representations of e). There is no upper limit on ℓ (since e.g. $(1\overline{1}1 \dots \overline{1})_2 = 1$), but low-weight B_m-representations never need be more than one digit longer than the binary representation: the bounds

$$\lfloor \log_2|e| \rfloor - \lfloor \log_2 m \rfloor \le \ell \le \lceil \log_2|e| \rceil$$

hold for the following well-known representations and for the newer representations that will be discussed afterwards.

– Let $w \ge 1$ be an integer parameter and $W = w + 1$. The *width-W nonadjacent form (W-NAF)* of e is a specific B_{2^w-1}-representation such that for $|e| \to \infty$, on average

$$\frac{\mathcal{H}(b_\ell \dots b_0)}{\lceil \log_2|e| \rceil} \approx \frac{1}{W+1}$$

assuming that e consists of random bits. As the W-NAF is a signed-digit equivalent of the well-known sliding-window technique for unsigned digits (cf. [4]), we also call it a *right-to-left signed window representation* with *window width* W.
(The origin of the 2-NAF is "property M" in [18]; the generalization to arbitrary $W \ge 2$ was alluded to in [20] and described independently in [12], in [22] as an improvement of a technique from [21], and in [2].)

– Now consider an arbitrary odd $m \ge 1$ and let

$$\begin{aligned} w_m &= \lfloor \log_2 m \rfloor + 1, \\ W_m &= w_m + 1, \\ \Delta_m &= \frac{2^{w_m} - 1 - m}{2^{w_m - 1}} \end{aligned}$$

(so that $0 \le \Delta_m < 1$). Generalizing right-to-left signed window representations, there is a *right-to-left signed fractional window (RL-SFW) representation* of e, the *m-RL-SFW* representation (details follow in Section 2). This is a B_m-representation such that for $|e| \to \infty$, on average

$$\frac{\mathcal{H}(b_\ell \dots b_0)}{\lceil \log_2|e| \rceil} \approx \frac{1}{W_m - \Delta_m + 1}$$

[1] For $\ell \ge 1$, an immediate optimization to the algorithm as written is to skip the first evaluation stage assignment and squaring if $b_\ell = 1$ (just keep g^2 in A from the precomputation stage) or $b_\ell = -1$ (just invert A to obtain g^{-2}).

assuming that e consists of random bits. If m is of the form $2^w - 1$ (so that $\Delta_m = 0$), the m-RL-SFW representation is the same as the W_m-NAF; otherwise the *effective window width* $W_m - \Delta_m$ is a fraction between w_m and W_m:

m	1	3	5	7	9	11	13	15	17	19	$\cdots$
$W_m - \Delta_m$	2	3	$3\frac{1}{2}$	4	$4\frac{1}{4}$	$4\frac{2}{4}$	$4\frac{3}{4}$	5	$5\frac{1}{8}$	$5\frac{2}{8}$	$\cdots$

(The RL-SFW representation was introduced in [14].)

The (finite-state) algorithms to obtain these representations given the binary representation of e read the least significant bit first and output the least significant signed digit first, proceeding towards the most significant input bit and output digit. This means they work *right to left* assuming big-endian notation; thus we speak of *RL-transformations.*

The use of an RL-transformation with LR-exponentiation means that usually the B_m-representation would be computed and stored before the actual exponentiation begins. This is unfortunate if memory is scarce. (Alternatively, the RL-transformation could be used repeatedly to determine the signed digits in the order in which they are needed, but this would mean an increased computational cost.) It is possible to perform *RL-exponentiation* instead so that the signed digits are used in the order in which they are generated (using algorithms from [23] and [11–exercise 4.6.3-9] as summarized in [14]); however, there are drawbacks:

- The group operation count to perform an RL-exponentiation is slightly higher than for an LR-exponentiation, given the same B_m-representation.
- The technique of employing mixed coordinates [3] for elliptic curves requires LR-exponentiation. (This technique uses additional precomputation effort to convert the elements G_b into a representation that accelerates evaluation stage operations $A \cdot G_b$ or $A \cdot G_{|b|}^{-1}$.)
- The technique of interleaved exponentiation [13] for efficiently computing power products $\prod_{1 \le j \le k} g_j^{e_j}$ applies to LR-exponentiation only.

Hence, left-to-right analogues of the above low-weight representations are called for.

A left-to-right analogue of the 2-NAF was described in [8], and recently, general left-to-right analogues of the signed window representation have appeared in [16], [1], and [17]. The latter two publications use an inherently identical LR-transformation, but describe it differently; see also [7–Section 6]. Also recently, proofs have appeared that the right-to-left signed window representation and its left-to-right variants are optimal in the sense of always achieving minimal weight ([15], [16], [1]): that is, given any e and w, no B_{2^w-1} representation $b'_{\ell'} \ldots b'_0$ of e can have lower weight than the $(w+1)$-NAF or its left-to-right analogues.

We generalize and extend these results by examining the right-to-left signed fractional window (RL-SFW) technique from [14] as well as its left-to-right variant (LR-SFW) implied by the approach of [17] and [7–Section 6]. (For *unsigned*

windows, right-to-left and left-to-right variants are equally simple: unsigned fractional windows, originally only presented for an RL-transformation in [14], have an immediate left-to-right analogue [5]. Signed-digit representations are trickier since they involve carries, but the approach of [17] and [7] makes it straightforward to come up with a left-to-right analogue of the RL-transformation from [14]; cf. [19] and [9].) We give minimality proofs for the weight of both RL-SFW and LR-SFW representations, and we examine the length of representations to study efficiency improvements beyond weight minimization. We always assume that e is positive: the case $e = 0$ is trivial; for negative e, apply the technique to $-e$ and replace all resulting digits by their negatives.

Section 2 looks at the RL-SFW representation and proves that it has minimal weight. Then Section 3 develops the LR-SFW representation and shows that it too has minimal weight. Section 4 points out that the left-to-right method is advantageous in that it tends to achieve slightly shorter lengths than the right-to-left method (and never yields a greater length), with details in Appendix A. It also considers how modified representations can further reduce the length in some cases; however, Appendix B observes that no finite-state transformation algorithm can always achieve minimal length among the representations of minimal weight.

2 Right-to-Left Signed Fractional Windows (m-RL-SFW)

The right-to-left signed fractional window (RL-SFW) representation was introduced (plainly as the "signed fractional window representation") in [14]. Here we describe the technique in a way that encompasses the non-fractional case as well (i.e. the right-to-left signed window representation, often called the W-NAF). Given any odd $m \geq 1$, let $w_m = \lfloor \log_2 m \rfloor + 1$ and $W_m = w_m + 1$; then we have $2^{w_m - 1} \leq m < 2^{w_m}$. The m-RL-SFW representation of any positive integer e is the B_m-representation $b_\ell \dots b_0$ obtained as follows.

First we define a mapping $\mathit{digit}_m \colon \{0, 1, \dots, 2^{W_m} - 1\} \to B_m \cup \{0\}$.

- If x is even, let $\mathit{digit}_m(x) = 0$;
- otherwise if $0 < x \leq m$, let $\mathit{digit}_m(x) = x$;
- otherwise if $m < x < 2^{W_m} - m$, let $\mathit{digit}_m(x) = x - 2^{w_m}$;
- otherwise (i.e. $2^{W_m} - m \leq x < 2^{W_m}$), let $\mathit{digit}_m(x) = x - 2^{W_m}$.

Observe that if x is odd, then $x - \mathit{digit}_m(x) \in \{0, 2^{w_m}, 2^{W_m}\}$. We extend the mapping to

$$\mathit{digit}_m \colon \mathbb{Z} \to B_m \cup \{0\}$$

by defining $\mathit{digit}_m(x) = \mathit{digit}_m(x \bmod 2^{W_m})$; it follows that $2^{w_m} \mid x - \mathit{digit}_m(x)$ for any odd $x \in \mathbb{Z}$. The RL-transformation algorithm in Fig. 2 on input any B_m-representation $b'_{\ell'} \dots b'_0$ of a positive integer e (e.g. the binary representation) generates a B_m-representation $b_\ell \dots b_0$ such that

$$b_i = \mathit{digit}_m \left(\frac{e - \sum_{0 \leq j < i} b_j \cdot 2^j}{2^i} \right).$$

{ **RL-transformation:** determine the
right-to-left signed fractional window (m-RL-SFW) **representation**
of $(b'_{\ell'} \dots b'_0)_2$ }

$i \leftarrow 0$
$D \leftarrow (b'_{w_m} \dots b'_0)_2$
while $D \neq 0 \ \vee \ i + w_m < \ell'$ **do**
  $d \leftarrow \mathit{digit}_m(D)$
  $b_i \leftarrow d$
  $i \leftarrow i + 1$
  $D \leftarrow b'_{i+w_m} \cdot 2^{w_m} + \dfrac{D - d}{2}$
$\ell \leftarrow i - 1$
return $(b_\ell, \dots, b_0)$

Fig. 2. RL-transformation for fractional windows

The algorithm assumes that $b'_i = 0$ for $i > \ell'$. It is easy to verify that the algorithm in Fig. 2 will always terminate with $b_\ell \dots b_0$ as given above, which implies $e = (b_\ell \dots b_0)_2$. (Note that

$$(b'_{\ell'} \dots b'_{i+W_m} \underbrace{0 \dots 0}_{w_m \text{ zeros}} d\, b_{i-1} \dots b_0)_2 = e$$

holds as a loop invariant.)

To see that the average weight for $|e| \to \infty$ with e composed of random bits satisfies

$$\frac{\mathcal{H}(b_\ell \dots b_0)}{\lceil \log_2 |e| \rceil} \approx \frac{1}{W_m - \dfrac{2^{w_m} - 1 - m}{2^{w_m - 1}} + 1} = \frac{1}{W_m + \dfrac{1+m}{2^{w_m-1}} - 1}$$

as claimed in Section 1, assume that the RL-transformation algorithm has to process an endless sequence of independently uniform random bits b'_i. Whenever the loop generates a non-zero digit b_i, the current value of D has its least significant bit set and is an otherwise random W_m-bit integer. Thus from the definition of digit_m it is clear that with probability $p = \dfrac{1+m}{2^{w_m}}$ we have $2^{W_m} \mid D - \mathit{digit}_m(D)$, and with probability $1 - p$ we have $D - \mathit{digit}_m(D) = 2^{w_m}$. In the latter case, the next non-zero output digit will follow after exactly $W_m - 2$ intermediate zeros; in the former case, the next non-zero output digit will follow after W_m intermediate zeros on average. This means that the total average a for the number of intermediate zeros is

$$a = pW_m + (1-p)(W_m - 2) = W_m + 2p - 2 = W_m + \frac{1+m}{2^{w_m-1}} - 2,$$

and thus the density $\dfrac{1}{a+1}$ of non-zero digits is as claimed above.

2.1 Minimality of Weight

To prove that the m-RL-SFW representation has minimal weight among all B_m-representations of any integer e, we show that

$$\mathcal{H}(b_\ell \dots b_0) \le \mathcal{H}(b'_{\ell'} \dots b'_0)$$

always holds if the transformation algorithm is applied to any B_m-representation $b'_{\ell'} \dots b'_0$ to obtain the corresponding RL-SFW representation $b_\ell \dots b_0$. For the analysis, we look at a variant of the algorithm from Fig. 2, shown in Fig. 3. This variant is easily seen to generate results that are identical except for leading zeros. The algorithm as written assumes that all variables b_i are initially zero.

{ **RL-transformation** (variant): determine the
right-to-left signed fractional window (m-RL-SFW) **representation**
of $(b'_{\ell'} \dots b'_0)_2$ }

$l \leftarrow \ell'$
$(b_l, \dots, b_0) \leftarrow (b'_{\ell'}, \dots, b'_0)$
$i \leftarrow 0$
while $i \le l$ **do**
  $\{ b_\ell, \dots, b_{i+1}, b_{i-1}, \dots, b_0 \in B_m \cup \{0\} \ \wedge \ |b_i| \le 2m \}$
  if b_i is even **then**
    $b_{i+1} \leftarrow b_{i+1} + b_i/2$
    $b_i \leftarrow 0$
    $\{ b_\ell, \dots, b_{i+2}, b_i, \dots, b_0 \in B_m \cup \{0\} \ \wedge \ |b_{i+1}| \le 2m \}$
    $i \leftarrow i + 1$
  else
    $D \leftarrow (b_{i+w_m} \dots b_i)_2$
    $d \leftarrow digit_m(D)$
    $(b_{i+w_m}, \dots, b_i) \leftarrow \left(\frac{D-d}{2^{w_m}}, \underbrace{0, \dots, 0}_{w_m - 1 \text{ zeros}}, d \right)$
    $\{ b_\ell, \dots, b_{i+w_m+1}, b_{i+w_m-1}, \dots, b_0 \in B_m \cup \{0\} \ \wedge \ |b_{i+w_m}| \le 2m \}$
    $i \leftarrow i + w_m$
  if $i > l \ \wedge \ b_i \ne 0$ **then**
    $l \leftarrow i$
$\ell \leftarrow i - 1$
return $(b_\ell, \dots, b_0)$

Fig. 3. RL-transformation (variant) for fractional windows

While the input and output consist only of digits from $B_m \cup \{0\}$, the variable b_i at the beginning of the loop body may contain other values; we call this digit the current *carry digit.* We can verify as a loop invariant that at the beginning of the loop body digits b_h other than the carry digit ($h \ne i$) will always

satisfy $b_h \in B_m \cup \{0\}$ (thus $|b_h| \leq m$) while the carry digit will always satisfy $|b_i| \leq 2m$. This clearly holds for $i = 0$. If for any i it holds at the beginning of the loop body, then it will also hold at the end of the loop body:

- If at the beginning of the loop body b_i is even, then it follows from $|b_{i+1}| \leq m$ and $|b_i| \leq 2m$ that
$$|b_{i+1} + b_i/2| \leq 2m.$$
- If at the beginning of the loop body b_i is odd, then
$$|D| \leq m \cdot 2^{w_m} + \ldots + m \cdot 2^1 + 2m = m \cdot 2^{w_m+1}$$
and thus
$$|D - d| \leq m \cdot 2^{w_m+1} + m.$$
Now because of $2^{w_m} \mid D - d$ and $m < 2^{w_m}$, it follows that
$$|D - d| \leq m \cdot 2^{w_m+1}$$
and thus
$$\left|\frac{D-d}{2^{w_m}}\right| \leq 2m.$$

So in both cases, the absolute value of the subsequent carry digit indeed cannot exceed $2m$. It is clear that no other digit will be set to values not in $B_m \cup \{0\}$.

Now we consider the value

$$\widetilde{H} = \mathcal{H}(b_l \ldots b_0) \; + \; \#\big\{h;\; |b_h| > m + 1\big\}$$

as observed at the beginning and at the end of the loop body (remember that b_i is the only digit among $b_l \ldots b_0$ that is not necessarily an element of $B_m \cup \{0\}$, so $\widetilde{H}$ exceeds the weight of $b_l \ldots b_0$ at most by one). Given the loop invariant, we can show that $\widetilde{H}$ will never increase within the loop body. It is clear that the loop body will not change any of the digits and thus not $\widetilde{H}$ if b_i is zero. If b_i is non-zero and even, following the algorithm it is easy to see that the changes done to b_i and b_{i+1} cannot increase $\widetilde{H}$. For b_i odd, at the beginning of the loop body define

$$H = \mathcal{H}(b_{i+w_m} \ldots b_i) \; + \; \#\big\{h;\; |b_h| > m + 1 \;\wedge\; i + w_m \geq h \geq i\big\};$$

now we can distinguish between the following cases:

- $H = 1$. This implies $b_i = D = d$, so the loop body will not change any of the digits and thus not $\widetilde{H}$.
- $H = 2$. If $|b_i| \leq m$, then there are initially exactly two non-zero digits among $b_{i+w_m} \ldots b_i$, both of absolute value at most m, and thus we have $|D| \leq (2^{w_m} + 1) \cdot m$. If $|b_i| > m$, then b_i is the only non-zero digit among $b_{i+w_m} \ldots b_i$, which implies $D = b_i$ and thus $|D| \leq 2m$. In both cases, for $d = \mathit{digit}_m(D)$ it follows that

$$\left|\frac{D-d}{2^{w_m}}\right| \leq \frac{(2^{w_m}+2)\cdot m}{2^{w_m}},$$

and since $2^{w_m} \mid D-d$ and $m < 2^{w_m}$,

$$\left|\frac{D-d}{2^{w_m}}\right| \leq \left\lfloor \frac{(2^{w_m}+2)\cdot m}{2^{w_m}} \right\rfloor = m + \left\lfloor \frac{2m}{2^{w_m}} \right\rfloor = m+1.$$

Thus when the loop body overwrites the digits $b_{i+w_m} \dots b_i$ with new values, the new carry digit will be of absolute value at most $m+1$; and since at most two of the new values will be non-zero, $\widetilde{H}$ cannot increase.

– $H \geq 3$. The digits $b_{i+w_m} \dots b_i$ are overwritten with new values out of which at most two are non-zero, and at most one is of absolute value larger than $m+1$; so these new digit values will contribute at most 3 to $\widetilde{H}$. This means that $\widetilde{H}$ cannot increase.

Initially, $\widetilde{H}$ is the input weight $\mathcal{H}(b'_{\ell'} \dots b'_0)$; in the end, it is the output weight $\mathcal{H}(b_\ell \dots b_0)$. $\widetilde{H}$ never increases, so no B_m-representation can have lower weight than the RL-SFW representation generated by the transformation algorithm.

3 Left-to-Right Signed Fractional Windows (m-LR-SFW)

To arrive at a left-to-right version, we use an approach from [17] and (building on [6] and [1]) from [7–Section 6]. This provides another way to view the RL-SFW method, and it yields an LR-SFW method (which was new at time of writing, but meanwhile has independently been described in [19] and [9].)

Given the binary representation $\beta_\lambda \dots \beta_0$ of any positive integer e, first let $\ell' = \lambda + 1$ and $b'_i = \beta_{i-1} - \beta_i$ for $i = \ell', \dots, 0$ (where $\beta_{-1} = 0$). Since

$$(b'_{\ell'} \dots b'_0)_2 \;=\; (\beta_\lambda \dots \beta_0\, 0)_2 - (\beta_\lambda \dots \beta_0)_2 \;=\; 2e - e \;=\; e,$$

this gives us a new B_1-representation $b'_{\ell'} \dots b'_0$ of e. Observe that this representation can be obtained from the binary representation just as easily in left-to-right as in right-to-left direction. It is clear from the construction of this new representation that every digit $b'_i = 1$ indicates that β_{i-1} is the left-most digit in a sequence of successive ones in the binary representation ($\beta_i = 0$, $\beta_{i-1} = 1$), and that every digit $b'_i = \overline{1}$ indicates that β_i is the right-most digit in such a sequence of ones ($\beta_i = 1$, $\beta_{i-1} = 0$). Thus, there must be an even number of non-zero digits in $b'_{\ell'} \dots b'_0$, and these show the following structure:

– The left-most non-zero digit is a 1.
– Skipping any zeros, the neighbors of a 1 digit will always have value $\overline{1}$ and the neighbors of a $\overline{1}$ digit will always have value 1.
– The right-most non-zero digit is a $\overline{1}$.

Because of this structure, we call $b'_{\ell'} \dots b'_0$ a *sign-alternating* B_1-representation. (This representation has previously been called "reversed binary representation" [10–exercise 4.1-27], "alternating greedy expansion" [6], and "mutual opposite form" [17].) If we write $*$ for any digit of value either 1 or $\overline{1}$, we get a simplified form from which the B_1-representation $b'_{\ell'} \dots b'_0$ can unequivocally be reconstructed due to its structure. We call $*$ and 0 *compressed digits.* The compressed-digit form of any subsequence of digits $b'_i \dots b'_h$ allows reconstructing the digits except for possible sign reversal (i.e. reconstruction would yield a sequence of digits that is identical to either $b'_i \dots b'_h$ or $\overline{b'_i} \dots \overline{b'_h}$).

The approach from [17] and [7] to obtain related RL- and LR-transformations is to apply a sliding-window technique to the sign-alternating B_1-representation: this can be done right to left, giving the well-known right-to-left signed window representation (W-NAF); or left to right, giving a left-to-right signed window representation. Generalizing this technique, we describe how a similar approach can be used with fractional windows. As before, let $w_m = \lfloor \log_2 m \rfloor + 1$ and $W_m = w_m + 1$ given an odd integer $m \geq 1$.

The sliding-window technique scans the representation $b'_{\ell'} \dots b'_0$ in one direction, either right to left *(RL-scanning)* or left to right *(LR-scanning),* starting a new window whenever a non-zero digit is encountered. Observe that for a window $\boxed{b'_{i+w} \dots b'_i}$ of any width $w+1$ in a sign-alternating B_1-representation, the window value $(b'_{i+w} \dots b'_i)_2$ will have absolute value at most 2^w, or $2^w - 1$ after dividing out powers of two. (In a B_1-representation that is not necessarily sign-alternating, the maximal absolute window value would be $2^{w+1} - 1$.) To accommodate fractional windows, the width of the current window is set to W_m if this is admissible, or w_m otherwise (or less than w_m when less than w_m digits are left for scanning). Here a window width is considered admissible if the window value is either some digit in B_m or even. (Window widths smaller than W_m are always admissible.) In any case, the window value will be the product of a power of two and a digit from B_m. Thus, each of the windows requires just one of the digits from B_m, appropriately positioned, to achieve the proper window value.

As an illustration of the transformations, we consider an example for $m = 5$. We have $w_m = 3$ and $W_m = 4$. Now windows of the form $\boxed{*\,0\,0*}$ and $\boxed{*\,0\,**}$ are not admissible while windows of the form $\boxed{**\,0\,*}$ and $\boxed{****}$ are admissible (because $(100\overline{1})_2 = (10\overline{1}1)_2 = 7 > m$ but $(1\overline{1}01)_2 = (1\overline{1}1\overline{1})_2 = 5 \leq m$, and similarly for the corresponding negative cases). Let $e = 22369 = (101011101100001)_2$; the sign-alternating B_1-representation of e obtained by the rule $b'_i = \beta_{i-1} - \beta_i$ is $1\overline{1}1\overline{1}100\overline{1}10\overline{1}0001\overline{1}$. RL-scanning does not encounter any inadmissible width-W_m windows; it yields the window constellation

$$\boxed{1}\boxed{\overline{1}\,1\,\overline{1}\,1}\,0\,0\,\boxed{\overline{1}\,1\,0\,\overline{1}}\,0\,\boxed{0\,0\,1\,\overline{1}},$$

resulting in the B_5-representation

$$\boxed{1}\boxed{0\,0\,0\,5}\,0\,0\,\boxed{0\,0\,0\,\overline{5}}\,0\,\boxed{0\,0\,0\,1}.$$

LR-scanning has to use width w_m (= 3) in one instance to avoid the inadmissible window $\boxed{1\,0\,0\,\bar{1}}$; it yields the window constellation

$$\boxed{1\,\bar{1}\,1\,\bar{1}}\,\boxed{1\,0\,0}\,\boxed{\bar{1}\,1\,0\,\bar{1}}\,0\,0\,0\,\boxed{1\,\bar{1}},$$

resulting in the B_5-representation

$$\boxed{5}\,\boxed{1\,0\,0}\,\boxed{0\,0\,0\,\bar{5}}\,0\,0\,0\,\boxed{0\,1}.$$

It is easy to see that this procedure with RL-scanning will always determine the same B_m-representation as the algorithms shown in Section 2 (ignoring any leading zeros); that is, this is just another way to view the RL-SFW technique. With LR-scanning, this is a new technique, giving us a *left-to-right signed fractional window (LR-SFW)* representation of e, the *m-LR-SFW* representation.

So far we have seen how to obtain LR-SFW representations using an intermediate sign-alternating B_1-representation (following [17] with appropriate changes for fractional windows). This intermediate step is helpful for describing and analyzing the method, but it is not necessary for implementation. Instead, the algorithm in Fig. 4 (following [1] with appropriate changes for fractional windows) can be used to obtain the m-LR-SFW representation $b_\ell \dots b_0$ of a positive integer directly from the binary representation $\beta_\lambda \dots \beta_0$. The algorithm as written assumes that $\beta_{\lambda+1} = 0$ and $\beta_i = 0$ for $i < 0$; also, all variables b_i are initially zero. In comments, we use b'_i as defined above ($b'_i = \beta_{i-1} - \beta_i$) to show that this algorithm expresses exactly the LR-transformation that we have introduced in terms of LR-scanning; to verify the correspondence, observe that for $i \geq h$

$$\begin{aligned}
(b'_i \dots b'_h)_2 &= (\beta_{i-1} \dots \beta_{h-1})_2 \;-\; (\beta_i \dots \beta_h)_2 \\
&= (\beta_{i-1} \dots \beta_h)_2 \cdot 2 \;+\; \beta_{h-1} \;-\; \beta_i \cdot 2^{i-h} \;-\; (\beta_{i-1} \dots \beta_h)_2 \\
&= -\beta_i \cdot 2^{i-h} \;+\; (\beta_{i-1} \dots \beta_h)_2 \;+\; \beta_{h-1} \\
&= (\overline{\beta_i}\beta_{i-1} \dots \beta_h)_2 \;+\; \beta_{h-1}.
\end{aligned}$$

3.1 Minimality of Weight

As discussed in Section 1, there are general advantages of LR-transformations over RL-transformations. A natural question to consider is whether despite of these advantages, the LR-SFW representation might be worse for exponentiation than the RL-SFW representation. To address this issue, here we show that for given m and e, the weight of the m-LR-SFW representation of e is the same as the weight of the m-RL-SFW representation. Thus, by the result from Section 2.1, the weight is minimal among all B_m-representations of e. (Later in Section 4 we will see that the LR-SFW representation actually has advantages beyond the general advantages of LR-transformations.)

Let ℓ' be a positive integer and

$$S = \left\{ s \in \{0, *\}^{\ell'} \mid * \text{ occurs an even number of times in } s \right\}$$

```
{ LR-transformation: determine the
  left-to-right signed fractional window (m-LR-SFW) representation
  of (β_λ ... β_0)_2 on binary input }

i ← λ + 1
ℓ ← 0
while i ≥ 0 do
    if β_i = β_{i−1} then     { b'_i = 0 }
        i ← i − 1
    else     { b'_i ≠ 0 }
        W ← W_m
        d ← (β̄_i β_{i−1} ... β_{i−W+1})_2 + β_{i−W}
        if d is odd ∧ |d| > m then
            W ← w_m
            d ← (β̄_i β_{i−1} ... β_{i−W+1})_2 + β_{i−W}
        { d = (b'_i ... b'_{i−W+1})_2 }

        next_i ← i − W
        i ← next_i + 1
        while d is even do
            i ← i + 1; d ← d/2
        { d is odd, |d| ≤ m, d = (b'_{next_i+W−1} ... b'_i)_2 }

        b_i ← d
        if i > ℓ then
            ℓ ← i
        i ← next_i
return (b_ℓ, ..., b_0)
```

Fig. 4. LR-transformation for fractional windows

the set of all compressed-digit forms of length ℓ'. We can examine the scanning process in terms of compressed-digit forms. For $s \in S$, let $\#LR_m(s)$ denote the number of windows that LR-scanning yields; i.e., $\#LR_m(s) = \mathcal{H}(b_\ell \dots b_0)$ where $b_\ell \dots b_0$ is the m-LR-SFW representation of the integer determined by the compressed-digit form s. Similarly, let $\#RL_m(s)$ denote the number of windows that RL-scanning yields. LR-scanning and RL-scanning are mostly symmetric, except for admissibility of window width W_m (for example, $(b'_{i+w_m} \dots b'_i)_2$ may be a digit in B_m when $(b'_i \dots b'_{i+w_m})_2$ is odd bot not in B_m). However, there is some symmetry that does respect admissibility: When a window is started and there are W_m compressed digits to look at (the first one of which in scanning direction is necessarily a $*$), there are 2^{w_m} possibilities what these W_m compressed digits might look like; and both for LR-scanning and for RL-scanning, window width W_m is admissible for exactly $m+1$ of these possibilities and not admissible for the remaining $2^{w_m} - m - 1$ possibilities. Thus, there is a bijection

$$\alpha \colon S \to S$$

such that the window structure (i.e., the positioning and width of windows, not taking into account the actual compressed digits in the windows) obtained by LR-scanning of any $s \in S$ is the exact mirror image of the window structure obtained by RL-scanning of $\alpha(s)$. This implies $\#LR_m(s) = \#RL_m\big(\alpha(s)\big)$.

Now assume that there was some specific $s_0 \in S$ such that $\#LR_m(s_0) \neq \#RL_m(s_0)$. By minimality of weight of the RL-SFW representation (Section 2.1), this would imply

$$\#LR_m(s_0) > \#RL_m(s_0),$$

i.e. $\#RL_m\big(\alpha(s_0)\big) > \#RL_m(s_0)$. Since α is a bijection, we have

$$\sum_{s\in S} \#RL_m\big(\alpha(s)\big) = \sum_{s\in S} \#RL_m(s),$$

so furthermore it would follow that there is some $s_1 \in S$ such that $\#RL_m\big(\alpha(s_1)\big) < \#RL_m(s_1)$. But this would mean

$$\#LR_m(s_1) < \#RL_m(s_1),$$

contradicting the minimality of weight of the RL-SFW representation. Thus no s_0 can exist for which $\#LR_m(s_0)$ differs from $\#RL_m(s_0)$.

4 Length

From Sections 2.1 and 3.1, we know that both the m-RL-SFW representation and the m-LR-SFW representation are optimal in the sense of having minimal weight among all B_m-representations. The efficiency of exponentiation given a B_m-representation $b_\ell \dots b_0$ with $b_\ell \neq 0$ depends not just on the weight $\mathcal{H}(b_\ell \dots b_0)$, but also on ℓ (see Section 1). Thus we are interested in representations that provide not only low weight, but also a short length $\ell+1$. Sometimes these goals are in conflict: for example, for B_7-representations of $255 = (1000000\overline{1})_2 = (70071)_2$, minimal weight and minimal length exclude each other; one or the other representation might provide better efficiency for LR-exponentiation depending on the relative speed of squarings and general multiplications in the group. We prioritize weight over length and consider only ways to reduce the length that do not increase the weight.

A first observation is that the LR-SFW representation can never be longer than the RL-SFW counterpart: Consider the scanning process on sign-alternating B_1-representations as described in Section 3, which yields a B_m-representation when each window value is expressed through a single non-zero digit from B_m. The maximal index ℓ of such a B_m-representation is the index of the right-most non-zero digit within the left-most window over the sign-alternating B_1-representation. For RL-scanning, the left-most (final) window will cover some number of non-zero digits of the sign-alternating B_1-representation, including its most significant digit. All of these non-zero digits, and possibly

more, would also be covered by the left-most (first) window obtained by LR-scanning. Thus, the maximal index ℓ cannot increase for LR-scanning compared with RL-scanning.

The example in Section 3 has already shown us that the m-LR-SFW representation is indeed shorter than the m-RL-SFW representation in some cases. In fact, the m-LR-SFW representation is advantageous for every m. For example, for $m = 1$, the average length saving for m-LR-SFW representations of long random integers e compared with m-RL-SFW representations is $1/6$; for $m = 3$, it is $1/2$; for $m = 7$, it is $37/40$. (Consider $m = 1$. If the left-most window over the sign-alternating B_1-representation is $\boxed{1\,0}$ or $\boxed{1}$, the 1-LR-SFW or 1-RL-SFW representation will be one digit longer than it is for the window $\boxed{1\,\overline{1}}$. The latter case happens with probability $1/2$ for LR-scanning, but only with probability about $1/3$ for RL-scanning. See Appendix A for more details.)

The m-LR-SFW representation does not guarantee minimal length among all B_m-representations of minimal weight. Substitution rules resulting in a modified (right-to-left) signed fractional window representation have been given in [14–Section 5.1], and these can similarly be applied to the left-to-right case to obtain shorter representations in some cases. Additional rules not mentioned in [14] are possible. For example, if $3 \le m \le 13$, the m-LR-SFW and m-RL-SFW representations both write $e = 15$ as $(1000\overline{1})_2$; this can be improved into $(303)_2$, or even into $(55)_2$ if $5 \le m$.

No LR-transformation implemented by a finite-state machine can always ensure minimal length among all minimal-weight B_m-representations; see Appendix B for examples. With the m-LR-SFW representation and its generally shorter length, there is less of a need to use modified representations to decrease the length than with the m-RL-SFW representation, in particular if m is relatively large. However, if program space permits, implementations can include a table of optimized substitution rules for certain prefixes that can be encountered in m-LR-SFW representations (such as $100\overline{1} \mapsto 31$ and $1000\overline{1} \mapsto 303$ for $m = 3$). While no such table could be complete for arbitrary lengths, this can help improve the average efficiency at least by a small margin.

Acknowledgement

Thanks to the anonymous reviewers for making me aware of [10–exercise 4.1-27], [6], and [7].

References

1. Avanzi, R. M. A note on the sliding window integer recoding and its left-to-right analogue. In *Selected Areas in Cryptography – SAC 2004*, Lecture Notes in Computer Science. To appear.

2. Blake, I. F., Seroussi, G., and Smart, N. P. *Elliptic Curves in Cryptography*, vol. 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1999.
3. Cohen, H., Ono, T., and Miyaji, A. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology – ASIACRYPT '98* (1998), K. Ohta and D. Pei, Eds., vol. 1514 of *Lecture Notes in Computer Science*, pp. 51–65.
4. Gordon, D. M. A survey of fast exponentiation methods. *Journal of Algorithms 27* (1998), 129–146.
5. Goriac, I., and Iftene, S. Personal communication, 2003.
6. Grabner, P. J., Heuberger, C., Prodinger, H., and Thuswaldner, J. M. Analysis of linear combination algorithms in cryptography. Preprint, 2003. Available from `http://www.opt.math.tu-graz.ac.at/~cheub/publications/`.
7. Heuberger, C., Katti, R., Prodinger, H., and Ruan, X. The alternating greedy expansion and applications to left-to-right algorithms in cryptography. Preprint, 2004. Available from `http://www.opt.math.tu-graz.ac.at/~cheub/publications/`.
8. Joye, M., and Yen, S.-M. Optimal left-to-right binary signed-digit recoding. *IEEE Transactions on Computers 49* (2000), 740–748.
9. Khabbazian, M., and Gulliver, T. A. A new minimal average weight representation for left-to-right point multiplication methods. Cryptology ePrint Archive Report 2004/266, 2004. Available from `http://eprint.iacr.org/`.
10. Knuth, D. E. *The Art of Computer Programming – Vol. 2: Seminumerical Algorithms*. Addison-Wesley, 1969.
11. Knuth, D. E. *The Art of Computer Programming – Vol. 2: Seminumerical Algorithms (3rd ed.)*. Addison-Wesley, 1998.
12. Miyaji, A., Ono, T., and Cohen, H. Efficient elliptic curve exponentiation. In *International Conference on Information and Communications Security – ICICS '97* (1997), Y. Han, T. Okamoto, and S. Qing, Eds., vol. 1334 of *Lecture Notes in Computer Science*, pp. 282–290.
13. Möller, B. Algorithms for multi-exponentiation. In *Selected Areas in Cryptography – SAC 2001* (2001), S. Vaudenay and A. M. Youssef, Eds., vol. 2259 of *Lecture Notes in Computer Science*, pp. 165–180.
14. Möller, B. Improved techniques for fast exponentiation. In *Information Security and Cryptology – ICISC 2002* (2003), P. J. Lee and C. H. Lim, Eds., vol. 2587 of *Lecture Notes in Computer Science*, pp. 298–312.
15. Muir, J. A., and Stinson, D. R. Minimality and other properties of the width-w nonadjacent form. *Mathematics of Computation*. To appear; preprint available from `http://www.cacr.math.uwaterloo.ca/tech_reports.html`.
16. Muir, J. A., and Stinson, D. R. New minimal weight representations for left-to-right window methods. In *CT-RSA 2005*, Lecture Notes in Computer Science. To appear; preprint available from `http://www.cacr.math.uwaterloo.ca/tech_reports.html`.
17. Okeya, K., Schmidt-Samoa, K., Spahn, C., and Takagi, T. Signed binary representations revisited. In *Advances in Cryptology – CRYPTO 2004* (2004), M. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, pp. 123–139.
18. Reitwiesner, G. W. Binary arithmetic. *Advances in Computers 1* (1960), 231–308.
19. Schmidt-Samoa, K., Semay, O., and Takagi, T. Analysis of some efficient window methods and their application to elliptic curve cryptosystems. Technical Report TI-3/04, 2004. Available from `http://www.informatik.tu-darmstadt.de/ftp/pub/TI/TR/`.

20. Schroeppel, R., Orman, H., O'Malley, S., and Spatscheck, O. Fast key exchange with elliptic curve systems. In *Advances in Cryptology – CRYPTO '95* (1995), D. Coppersmith, Ed., vol. 963 of *Lecture Notes in Computer Science*, pp. 43–56.
21. Solinas, J. A. An improved algorithm for arithmetic on a family of elliptic curves. In *Advances in Cryptology – CRYPTO '97* (1997), B. S. Kaliski, Jr., Ed., vol. 1294 of *Lecture Notes in Computer Science*, pp. 357–371.
22. Solinas, J. A. Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography 19* (2000), 195–249.
23. Yao, A. C.-C. On the evaluation of powers. *SIAM Journal on Computing 5* (1976), 100–103.

A LR-SFW versus RL-SFW: Length Comparison

We want to compare the expected lengths of left-to-right and right-to-left signed fractional window representations of long random integers by looking at LR-scanning and RL-scanning as described in Section 3. Assume an ℓ'-bit integer e is given (so that $b'_{\ell'} = 1$ in the sign-alternating B_1-representation). By the probabilities below, in the m-LR-SFW representation the expected maximum index ℓ is $\ell' - 1/2$ for $m = 1$, $\ell' - 5/4$ for $m = 3$, and $\ell' - 17/8$ for $m = 7$; in the m-RL-SFW representation it is about $\ell' - 1/3$ for $m = 1$, about $\ell' - 3/4$ for $m = 3$, and about $\ell' - 6/5$ for $m = 7$.

Consider $m = 2^w - 1$ such that windows over the sign-alternating form have width $w + 1$. For LR-scanning over random compressed-digit forms of sufficient length, the left-most window is

- $\boxed{*0\ldots0}$ with probability 2^{-w};
- $\boxed{**0\ldots0}$ also with probability 2^{-w};
- of the form $\boxed{*?*0\ldots0}$ with probability 2^{1-w};
- $\ldots$;
- of the form $\boxed{*?\ldots?*}$ with probability 2^{-1}.

As the resulting B_m-representations will be successively shorter (the maximal index ℓ of the B_m-representation is the index of the right-most non-zero digit within the left-most window over the compressed-digit form), this is in favor of generating short B_m-representations. No $m \geq 2^w - 1$ will have average lengths longer than this.

For RL-scanning over long random compressed-digit forms, however, the left-most window is

- $\boxed{*}$ (width 1) with probability about $2/(w + 2)$;
- $\boxed{**}$ (width 2) with probability about $1/(w + 2)$;
- of the form $\boxed{*?*}$ (width 3) with probability about $1/(w + 2)$;
- $\ldots$;
- of the form $\boxed{*?\ldots?*}$ (width $w + 1$) with probability about $1/(w + 2)$.

Again the resulting B_m-representations will be successively shorter, but here the probabilities are in favor of generating long B_m-representations. No $m \le 2^w - 1$ will have average lengths shorter than this. (To derive these estimates, assume that RL-scanning is applied to a long sequence of independently uniformly random compressed digits. Any given $*$ in a width-w window well within such a sequence, after earlier windows have provided plenty of randomization of window positions, will be at the right-most position of its window with probability about $2/(w+2)$, and at any other position of its window with probability about $1/(w+2)$ each. This is seen by counting how many of the 2^w possibilities for the content of a width-$(w+1)$ window in RL-scanning have a $*$ in the respective position: the right-most position sees twice the proportion of $*$'s as each other position. Since the start of a window does not depend on what follows, the left-most $*$ in a [finite but long] compressed-digit form is similar except that the window abruptly ends there.)

B Limitations of Length Reduction

Let $\langle m \rangle$ be a shorthand notation for a digit string consisting of $w_m - 1$ zeros concatenated with the digit m, and $\langle \cancel{m} \rangle$ for w_m zeros (e.g. $\langle 5 \rangle = 005$, $\langle \cancel{5} \rangle = 000$). Now consider the integers with B_m-representations either of the form

$$m1\ 0\langle m \rangle\ 0\langle m \rangle \ldots 0\langle m \rangle\ 0\langle m \rangle\ \langle m \rangle$$

or of the form

$$m1\ 0\langle m \rangle\ 0\langle m \rangle \ldots 0\langle m \rangle\ 0\langle m \rangle\ \langle \cancel{m} \rangle.$$

The m-LR-SFW representation for an integer of the first form is longer than the above representation and has lower weight; this is a weight minimization that would not be possible without the length increase. For example, for $m = 1$,

$$(11\,01\,01 \ldots 01\,01\,1)_2$$

is rewritten with lower weight as

$$(100\,\overline{1}0\,\overline{1}0 \ldots \overline{1}0\,\overline{1}0\,\overline{1})_2.$$

The m-LR-SFW representation for an integer of the second form is longer as well, but the weight remains unchanged; the original representation as given above already has minimal weight. For example,

$$(11\,01\,01 \ldots 01\,01\,0)_2$$

has no B_1-representation of lower weight. There is no bound on the number of digits that one might have to examine to distinguish between such cases, so no finite-state machine could always generate the shortest representation among those of minimal weight.