The Montgomery and Joye Powering Ladders are Dual

Colin D. Walter

Information Security Group,
Royal Holloway University of London,
Egham, Surrey, TW20 0EX, United Kingdom.

email CDW.iacr@gmail.com

Abstract

Hitherto the duality between left-to-right and right-to-left exponentiation algorithms has been a loosely defined concept. Recently, the author made the definition precise by adding requirements on space usage and operation types. Here it is shown that the Montgomery and Joye powering ladders are dual in this sense. Several versions of these algorithms are derived naturally with a cost-free, natural, built-in blinding mechanism as a side channel counter-measure.

Key Words: Algorithm Theory, Exponentiation, Cryptography, Elliptic Curve Cryptography, Public Key Cryptography, Scalar Multiplication.

1 Introduction

Cryptography in resource-constrained devices such as smart cards requires computation which is ideally space efficient, time efficient, leak resistant and fault resistant. This has stimulated the development of exponentiation (or scalar multiplication) algorithms as these are vital for most public key cryptosystems.

In [13], the author defined a duality between left-to-right and right-to-left methods which is a tighter version of the transposition method [8, 3]. For the first time this enables both space *and* time properties to be transferred systematically when switching the direction of processing exponent digits. [13] shows that the m-ary algorithms of Brauer and Yao [4, 14] are dual in this stronger sense. Their base-2 cases are the widely used left-to-right and right-to-left binary square-and-multiply methods. Here we contribute to the understanding of exponentiation by studying the important powering ladders of Montgomery [5, 9, 10] and Joye [6, 7], showing that they too are dual.

2 Space Duality

There are two key ingredients to the duality. The first is to restrict the allowable operations. A *location-aware* addition chain is a finite sequence of the following operations ([13], Defn. 2):

i) Copying from one named register to another.
ii) Copying from one named register to another combined with initialising the source register to the group identity.

iii) Multiplying the contents of two distinct named registers and writing the result into one of those registers.
iv) Multiplying the contents of two distinct named registers, writing the product into one of those registers and initialising the other register to the group identity.
v) Raising the contents of a register to the power s for some $s \in \mathbb{Z}$.
vi) Swapping the contents of two named registers.

(For most cryptographic applications the group identity is either $1 \in \mathbb{N}$ for integer-based cryptosystems or the point $\mathcal{O}$ at infinity on an elliptic curve.)

The other key ingredient is to write the algorithm following several rules which are natural for efficiency and good housekeeping. A location-aware chain is *normalised* if it satisfies the following (cf [13], Defn. 6):

i) There is a prescribed subset of registers used for I/O. Each such register reads in an external value *and* writes a value to output. No other registers make use of external communication.
ii) All initial values in non-I/O registers are assumed to be undefined, and all initial values in I/O registers must be well-defined and used by some operation.
iii) All final values in non-I/O registers must be explicitly set to the group identity element by an operation with initialisation, unless the register has not been used at all. No I/O register must have a final value which has been explicitly set to the identity element.
iv) The inputs of every operation must be well-defined and not include a value that has been set to the identity element by an operation with initialisation. The initial values of I/O registers and the non-initialised outputs of operations must be used by a subsequent operation or be the final value of an I/O register.

Generally, any exponentiation algorithm can be expressed easily in the above way although sometimes composite operations (e.g. $x, y \mapsto x^2 y$) may need to be artificially broken into parts that would not be implemented separately. Moreover, an implementation might often skip some processes such as setting some final register values to the identity.

The *dual* of a normalised location-aware chain is the normalised location-aware chain given by applying the transpose operation. If each of its operations is described using a matrix which is applied to the vector of register values, then the dual is given by matrix transposition. The transpose of any of the above chain operations also represents an allowable chain operation. Although some copying operations become multiplications and *vice versa* as a result of this process, the numbers of multiplications and copyings are unchanged when taking the dual. However, the swap and powering operations remain as swap and powering operations under transposition.

3 Two Montgomery Ladders

The original Montgomery Powering Ladder [10] is a left-to-right exponentiation algorithm. A slightly modified version for a multiplicative group G is given in Fig. 1. The algorithm uses two registers for calculations and so really computes two exponentiations. This provides the opportunity to introduce a blinding parameter g_0 and to output a second value. Taking $g_0 = 1_G$ to be the group identity yields the usually required output ${g_1}^D$, but otherwise there are standard ways of dealing with the blinding factor ${g_0}^{\overline{D}}$ such as computing it once, and thereafter squaring both g_0 and ${g_0}^{\overline{D}}$ after every exponentiation. The outputs differing by the constant factor $g_0 g_1^{-1}$ provides an easy check on whether an attacker has induced a fault in an attempt to recover the secret exponent D. The last

aspect of generalisation is to drop the requirement for the leading bit of D being 1. The correctness proof is by induction on $i = n-1, ..., 1, 0$ using $D_i = \sum_{j=i}^{n-1} d_j 2^{j-i}$, $D_{i-1} = 2D_i + d_{i-1}$ and $D_i' = 2^{n-i} - D_i$ with the additional loop invariant $T[0]/T[1] = g_0 g_1^{-1}$.

Inputs: $g_0, g_1 \in G$, $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i \in \{0, 1\}$ for $0 \leq i < n$.
Output: ${g_0}^{\overline{D}} {g_1}^{D}$, ${g_0}^{\overline{D+1}} {g_1}^{D+1}$ where $D + \overline{D} = 2^n$.

$T[0] \leftarrow g_0$
$T[1] \leftarrow g_1$
[$C \leftarrow T[0]/T[1]$] // optional – see §4
for $i \leftarrow n-1$ **down to** 0 **do** {
    $T[1-d_i] \leftarrow T[0] \times T[1]$
    $T[d_i] \leftarrow T[d_i]^2$ }
return $T[0]$, $T[1]$

Figure 1: A Left-to-Right Montgomery Powering Ladder with Blinding

Inputs: $g_0, g_1 \in G$, $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i \in \{0, 1\}$ for $0 \leq i < n$.
Output: ${g_0}^{\overline{D}} {g_1}^{\overline{D+1}}$, ${g_0}^{D} {g_1}^{D+1}$ where $D + \overline{D} = 2^n$.

$T[0] \leftarrow g_0$
$T[1] \leftarrow g_1$
for $i \leftarrow 0$ **to** $n-1$ **do** {
    $T[d_i] \leftarrow T[d_i]^2$
    $T[d_i] \leftarrow T[0] \times T[1]$ }
return $T[0]$, $T[1]$

Figure 2: The Dual Right-to-Left Powering Algorithm with Blinding

Taking the transpose of each multiplicative operation and reversing their order yields the dual right-to-left algorithm given in Fig. 2. It clearly has the same number of squarings and multiplications as the left-to-right version. However, it has different outputs. The transformation achieved by the left-to-right ladder can be described via the matrix

$$\begin{bmatrix} \overline{D} & D \\ \overline{D+1} & D+1 \end{bmatrix}$$

and so the dual computes values corresponding to the transpose of this. Then the exponent D appears in the output value of $T[1]$ rather than $T[0]$, and g_0 becomes the parameter which is raised to the power D, with g_1 now providing the blinding. The special case $g_1 = 1_G$ reduces the algorithm to Joye's Algorithm 3 in [6] (with the registers interchanged). Thus Joye's right-to-left algorithm has a very natural derivation as the dual of Montgomery's left-to-right powering ladder.

A striking difference between the algorithms is that the left-to-right version could perform the two operations of a loop iteration in parallel, whereas those of the dual algorithm must be performed sequentially.

4 A Revised Right-to-Left Algorithm

Both the algorithms above take n squarings and n multiplications to complete the computation, thereby adding about one third to the number of multiplicative operations when compared with standard binary exponentiation. However, because both multiplications in a loop iteration share a common multiplicand in the left-to-right version, there are considerable savings to be made in that case (*see* [5], §3.3). Alternatively, in the Weierstraß elliptic curve context, the main loop of the left-to-right version can be executed entirely in terms of the x-coordinate, from which the y-coordinate of the final output is easily computed [1, 9]. This efficiency gain makes the Montgomery ladder competitive with the normal square-and-multiply algorithm. Specifically, (using additive notation for G), if the x-coordinates of $P \neq \mathcal{O}$, $Q \neq \pm P$ and $P-Q$ are known, then there are straightforward, explicit formulae for the coordinates of $2P$ and $P+Q$ in terms of them [10]. Also, when $Q \neq \mathcal{O}$ there is an explicit formula for the y-coordinate of P in terms of those three x-coordinates [1, 9, 11, 12, 2]. So, in the main loop, if the x-coordinates of points $T[0]$, $T[1]$ and $T[0]-T[1]$ are known at the start of an iteration, then the x-coordinates for $T[0]$ and $T[1]$ at the end of the iteration can be computed. And, since $C = T[0]-T[1]$ is a loop invariant, its x-coordinate is also known after computing it once at initialisation. Finally, the formulae for the y-coordinate can be used at the end of the loop for determining in full each of the desired final output points.

At first sight, a similar efficiency gain seems impossible for the right-to-left direction since the correct analogue of $T[0]-T[1]$, namely $T[1]+T[0]$, is, unfortunately, *not* a loop invariant. However, suppose the loop iteration starts with known x-coordinates for P, Q and $P+Q$, where P is the point associated with $T[d_i]$ and Q with $T[1-d_i]$. It is required to compute the x-coordinates for $2P+Q$ (the new value for $T[d_i]$) and $2P+2Q$ (the new value for $T[0]+T[1]$), but not that of $2P$ which is just the temporary value of $T[d_i]$ in the middle of the iteration. The doubling of $P+Q$ is easy using the given formula, and $2P+Q$ simply requires using the three x-coordinates of $P, P+Q, Q = (P+Q)-P$ to determine the x-coordinate for $P+(P+Q)$. The third x-coordinate required for the start of the next iteration is that of Q, which is already known. So the three x-coordinates are ready for the following iteration or for computing the y-coordinate at the end. The version of Fig. 2 modified in this way to include $P+Q$ in a new variable C is given in Fig. 3 (*cf* Joye [7] Alg. 7).

Inputs: $g_0, g_1 \in G$, $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i \in \{0,1\}$ for $0 \leq i < n$.
Output: ${g_0}^{\overline{D}} {g_1}^{\overline{D+1}}$, ${g_0}^{D} {g_1}^{D+1}$ where $D + \overline{D} = 2^n$.

$T[0] \leftarrow g_0$
$T[1] \leftarrow g_1$
$C \leftarrow T[0] \times T[1]$
for $i \leftarrow 0$ **to** $n-1$ **do** {
    $T[d_i] \leftarrow T[d_i] \times C$
    if $i \neq n-1$ **then** $C \leftarrow C^2$ }
return $T[0]$, $T[1]$

Figure 3: Revised Right-to-Left Exponentiation Algorithm

As in the left-to-right version of Fig. 1, there is one application of each x-coordinate

formula per iteration – one for the addition and one for the doubling. So, with both using C, the speed and space usage should be similar for both directions of the Montgomery Ladder. The two loop multiplications can again be done in parallel as Fig. 1 and their common multiplicand could lead to savings in other contexts, as noted above.

If one entirely omits the updating of $T[0]$ within the loop and considers just the updating and output for $T[1]$, then Fig. 3 is reduced almost to just the normal right-to-left binary (square-and-multiply) algorithm.

5 The Dual Left-to-Right Algorithm

Taking the dual of Fig. 3 yields another derivative of the left-to-right Montgomery Ladder, using three registers rather than two. This is given in Fig. 4 (*cf* Joye [7] Alg. 6). Its disadvantages include the loss of processing using only x-coordinates in elliptic curve applications, and the loss of parallel processing in the loop. Its main advantage, shared with Fig. 3, is the ease of generalising it to an m-ary algorithm, as in Joye's Algorithms 4 and 5 [7]. The odd-looking final two multiplications are the result of the dualising process.

Inputs: $g_0, g_1 \in G$, $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i \in \{0, 1\}$ for $0 \leq i < n$.
Output: ${g_0}^{\overline{D}} {g_1}^{D}$, ${g_0}^{\overline{D+1}} {g_1}^{D+1}$ where $D + \overline{D} = 2^n$.

$T[0] \leftarrow g_0$
$T[1] \leftarrow g_1$
$C \leftarrow 1_G$
for $i \leftarrow n{-}1$ **down to** 0 **do** {
  if $i \neq n{-}1$ **then** $C \leftarrow C^2$
  $C \leftarrow T[d_i] \times C$ }
$T[0] \leftarrow T[0] \times C$
$T[1] \leftarrow T[1] \times C$
return $T[0]$, $T[1]$

Figure 4: Another Left-to-Right Binary Exponentiation with Blinding

For $g_0 = 1_G$, $T[0] = 1_G$ always and so the loop multiplication of Fig. 4 does nothing when processing a zero bit. Then the algorithm is essentially just the usual left-to-right binary square-and-multiply algorithm, derived naturally from Mongomery's Powering Ladder.

6 Final Remarks

All four of the algorithms presented have properties which are useful against side channel and fault attacks. They can be classified as "square and multiply always" and are "highly regular" in the sense of Joye [6], i.e. the same sequence of operations is always executed independently of the data. Moreover, if both outputs are taken, they have no dummy operations.

The main loops have been the primary concern here. Inputs, outputs, and assignments outside the main loops have been stripped almost to a minimum, but can be easily adapted to particular applications.

7 Conclusion

The power of duality between exponentiation algorithms has been illustrated, providing deeper insight into powering ladder algorithms, and how they are related to each other and to standard binary exponentiation algorithms. Duality has led to generalisations of them which provide base point blinding. Thus, duality is a valuable tool for developing and investigating existing and new exponentiation algorithms.

References

[1] Agnew, G. B., Mullin, R. C., and Vanstone, S. A., *An Implementation of Elliptic Curve Cryptosystems Over $F(2^{155})$*, IEEE Journal on Selected Areas in Communications, June 1993, **11** (5), pp. 804–813.

[2] Brier, E. and Joye, M., *Weierstraß Elliptic Curves and Side-Channel Attacks*, PKC 2002, LNCS **2274**, Springer-Verlag, 2002, pp. 335–345.

[3] Bernstein, D. J., *Pippenger's Exponentiation Algorithm*, http://cr.yp.to/papers/pippenger.pdf, 2002.

[4] Brauer, A.: *On Addition Chains*, Bull. Amer. Math. Soc., 1939, **45** (10), pp. 736–739.

[5] Joye, M., and Yen, S.-M., *The Montgomery Powering Ladder*, CHES 2002, LNCS **2523**, Springer-Verlag, 2003, pp. 291–302.

[6] Joye, M., *Highly Regular Right-to-Left Algorithms for Scalar Multiplication*, CHES 2007, LNCS **4727**, Springer-Verlag, 2007, pp. 135–147.

[7] Joye, M., *Highly Regular m-Ary Powering Ladders*, Selected Areas in Cryptography, LNCS **5867**, Springer-Verlag, 2009, pp. 350–363.

[8] Knuth, D. E., *The Art of Computer Programming*, vol. **2**, "Seminumerical Algorithms", §4.6.3, 3rd Edition, Addison-Wesley, 1998, pp. 465–485.

[9] López, J., and Dahab, R., *Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation*, CHES 1999, LNCS **1717**, Springer-Verlag, 2000, pp. 316–327.

[10] Montgomery, P. L., *Speeding the Pollard and Elliptic Curve Methods of Factorization*, Maths. of Computation, Jan. 1987, **48** (177), pp. 243–264.

[11] Okeya, K., Kurumatani, H., and Sakurai K., *Elliptic Curves with the Montgomery Form and their Cryptographic Applications*, Public Key Cryptography, LNCS **1751**, Springer-Verlag, 2000, pp. 238–257.

[12] Okeya, K., and Sakurai, K., *Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-Coordinate on a Montgomery Form Elliptic Curve*, CHES 2001, LNCS **2162**, Springer-Verlag, 2001, pp. 126–141.

[13] Walter, C. D., *A Duality in Space Usage between Left-to-Right and Right-to-Left Exponentiation*, CT-RSA 2012, LNCS **7178**, Springer-Verlag, 2012, pp. 84–97.

[14] Yao, A. C.-C., *On the Evaluation of Powers*, SIAM J. Comput., 1976, **5** (1), pp. 100–103.