

Research Report


On Sliding Window Exponentiation

L. O'Connor

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

 Research
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

On Sliding Window Exponentiation

L. O'Connor

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

The sliding window method is a general purpose algorithm to determine the exponentiation g^e in a general group. Let $s_{nk}(e)$ and $m_{nk}(e)$ denote, respectively, the number of squarings and multiplications required by the sliding window method when g^e is computed using k -bit windows, and e is a uniformly distributed n -bit exponent. In this paper we prove that $\mathbf{E}[s_{nk}(e)] = n - k + O(1)$ and $\mathbf{E}[m_{nk}(e)] = 2^{k-1} + n/(k + 1) + O(1)$, and further that both distributions are concentrated around their respective means.

Keywords : Exponentiation; Cryptography; Formal languages; Combinatorial problems.

1 Introduction

The sliding-window method for (modular) exponentiation is a variant of the well-known b -ary method [8], and is the ‘recommended method’ for general exponentiation [9, p.617]. When $b = 2^k$, the 2^k -ary method can be considered as parsing an exponent e into adjacent windows of k -bits, where the least significant window may cover less than k bits. The idea of the sliding-window method is to select the placement of each k -bit window so that its most and least significant bit are equal to one. The advantage of such a partition over the 2^k -ary method is that (1) the number of windows is expected to be reduced as runs of zeroes may occur between consecutive windows, and (2) the amount of precomputation is reduced as the windows only represent odd powers. The precomputation requires one squaring and $2^{k-1} - 1$ multiplications [9, p.616], while the main computation loop requires $w_{nk}(e) - 1$ multiplications, where $w_{nk}(e)$ is the number of windows produced by the window partition. The exact number of squarings will depend on k and the exponent length n , but will be between $(n - 1) - (k - 1)$ and $n - 1$. The sliding-window method for k -bit windows is shown in Figure 1.

Though the sliding window method has been used by practitioners since at least the early 80’s [13], it did not begin to appear in the literature until later. The method is alluded to by Bos and Coster [1], and we find the algorithm being reinvented a few years later by Hui and Lam [6]. There is no mention of the sliding window method in Knuth [8], and some authors still invent ‘improved’ exponentiation algorithms without reference to the method (see [12] for example).

References to analyses of the sliding window method appear to be scattered, and not widely known. The average number of windows, $\mathbf{E}[w_{nk}(e)]$, is approximately $n/(k + 1)$ since the average length of the precomputed exponents is $(k - 1)$ bits, and on average a window will be followed by approximately two zeros. Further, computational results from sampling in the range of 512 – 1024 bits, even for a relatively small number of exponents, reveals the trend that the number of windows is concentrated around $n/(k + 1)$. Thus it is now common to find references stating that the k -bit sliding window method requires $n/(k + 1)$ windows on average (see [4, 2]). However no average case analysis of $w_{nk}(e)$ was reported in [9, p.617]. The first attempted analysis (we found) is due to Hui and Lam [6] who gave a recurrence for $\mathbf{E}[w_{nk}(e)]$ without proof or solution. Shortly after, Kóc [7] analysed two variants of the partitioning method given in Figure 1 using Markov chains, and presented several tables of computational results for various parameter choices. A thorough analysis of various average case parameters

```

Precompute  $g^{2^{i-1}}$ ,  $2 \leq i \leq 2^{k-1}$  ;
 $X \leftarrow 1$ ;  $i \leftarrow (n - 1)$  ;
while  $i \geq 0$  do
    if  $e_i = 0$  then
         $X \leftarrow X^2$ ;  $i \leftarrow i - 1$  ;
    else
        Find the longest string  $e_i e_{i-1} \cdots e_t$  such that  $i - t + 1 \leq k$  and  $e_t = 1$ 
         $X \leftarrow X^{2^{i-t+1}} \cdot g^{e_i e_{i-1} \cdots e_t}$ ;  $i \leftarrow t - 1$  ;
    fi
od
output  $X$  ;

```

Figure 1: The sliding-window exponentiation method.

for the sliding window method was given by Cohen [3] for n -bit exponents e where the most significant bit is one (that is $2^{n-1} \leq e < 2^n$). No reference in [3] was made to either [6] or [7].

In this paper we extend the work of [10] with some results of [3] to determine the expectation and variance of $s_{nk}(e)$ and $m_{nk}(e)$ when e is a uniformly distributed n -bit exponent. We show that both distributions are concentrated around their expectations, where $\mathbf{E}[s_{nk}(e)] = n - k + O(1)$ and $\mathbf{E}[m_{nk}(e)] = 2^{k-1} + n/(k+1) + O(1)$. Throughout the paper we assume that e is n bits in length, and that the sliding window method uses k bit windows, $k \geq 2$. To emphasize these conditions $s(e)$, $m(e)$ and $w(e)$ should be written as $s_{nk}(e)$, $m_{nk}(e)$ and $w_{nk}(e)$ but we will use the former notation for simplicity, and hope that no confusion arises.

2 The multiplication distribution

Since $m(e) = 2^{k-1} + w(e) - 2$ we may determine the distribution of $m(e)$ from the distribution of $w(e)$. Let the generating function $G_k(x, z)$ be defined as

$$G_k(x, z) = \sum_{n, m \geq 0} a_{nm} x^n z^m = \sum_{n, m \geq 0} (2^n \cdot \Pr(w(e) = m)) x^n z^m, \quad (1)$$

so that a_{nm} is the number of n -bit binary strings e for which $w(e) = m$.

Theorem 2.1 If $G_k(x, z) = \sum_{n, m \geq 0} a_{nm} x^n z^m$ then

$$G_k(x, z) = \frac{1 - 2x + zx - zx^k 2^{k-1}}{(1 - x - zx^k 2^{k-1})(1 - 2x)}. \quad (2)$$

Proof. Consider the following regular expression

$$R_k = R_1^* R_2 = \left(0 + 10^{k-1} + \sum_{i=0}^{k-2} 1(0+1)^{k-2-i} 10^i \right)^* \left(\epsilon + \sum_{i=0}^{k-2} 1(1+0)^i \right).$$

R_1 generates the single word 0 and all words of length k that start 1. Then R_1^* generates all words corresponding to k -bit windows separated by runs of zeroes. R_2 generates either the empty string or a word beginning with 1, of length less than k , which corresponds to the case where $W_{w(e)}$ is less than k bits in length.

We now mark R_1 for length and weight as follows: 0 is marked x , 10^{k-1} is marked zx^k meaning it has length k and corresponds to one nonzero window in the parse of the exponent, and $1(0+1)^{k-2-i} 10^i$ is similarly marked as $zx^2(x+x)^{k-2-i}$. Using the same rules for R_2 we have that

$$\begin{aligned} G_{R_1}(x, z) &= x + zx^k + zx^2 \sum_{i=0}^{k-2} (x+x)^{k-2-i} = x + zx^k + zx^k 2^{k-2} (2 - 2^{2-k}), \\ G_{R_2}(x, z) &= 1 + zx \left(\frac{1 - x^{k-1} 2^{k-1}}{1 - 2x} \right). \end{aligned}$$

Since each substring in R_1 and R_2 is unique then $G_{R_1}(x, z)$ and $G_{R_2}(x, z)$ are the generating functions for length and weight of R_1 and R_2 respectively. We need only derive the generating function for R_1^* . We observe that if e' is any string generated by R_1^* then there is only one way to combine the words of R_1 to produce e' . Hence R_1^* is said to be *unambiguous* and it is known [11, p.378] that given $G_{R_1}(x, z)$ enumerates R_1 then $1/(1 - G_{R_1}(x, z))$ enumerates R_1^* . The theorem follows from simplifying $G_k(x, z) = 1/(1 - G_{R_1}(x, z)) \cdot G_{R_2}(x, z)$. \square

As a check on our derivation, we observe that $G_k(x, 1) = 1/(1 - 2x)$, the generating function for length of all binary strings. For computational purposes, $G_k(x, z)$ can be expanded as a power series and the distribution of $w(e)$, and hence $m(e)$, examined. We summarize the distribution of $m(e)$ through its expectation and variance in the following corollary.

Corollary 2.1 Let $m(e) = m_{nk}(e)$ be the number of multiplications required by the sliding window method to compute g^e for the n -bit exponent e using k -bit windows. Then for a uniformly distributed n -bit exponent e

$$\mathbf{E}[m_{nk}(e)] = 2^{k-1} - 2 + \frac{n}{(k+1)} + \frac{k(k-1)}{2(k+1)^2} + o(1), \quad (3)$$

$$\mathbf{Var}[m_{nk}(e)] = \frac{2n}{(k+1)^3} + \frac{k(k^3 + 4k^2 - 19k - 18)}{4(k+1)^4} + O(1). \quad (4)$$

Proof. We will prove the corollary by considering the expectation and variance of $w(e)$. To this end, let $F_k(x)$ be defined as $F_k(x) = \sum_{n \geq 0} \mathbf{E}[w(e)]x^n$ where

$$F_k(x) = G'_k(x/2, 1) = \left. \frac{\partial G_k(x/2, z)}{\partial z} \right|_{z=1} = \frac{x}{(1-x)(2-x-x^k)}. \quad (5)$$

We first note that since

$$(2-x-x^k) = (1-x) \left(1 + \frac{1-x^k}{1-x} \right) \stackrel{\text{def}}{=} (1-x)p_k(x), \quad (6)$$

the partial fraction expansion of $F_k(x)$ then has the form

$$F_k(x) = \frac{A}{(1-x)^2} + \frac{B}{(1-x)} + \frac{q(x)}{p_k(x)} \quad (7)$$

where $q(x)$ is a polynomial of degree less than $(k-1)$. It is known [3, Lemma 2.2] that the $(k-1)$ roots of the reflected polynomial [5, p.325] associated with $p_k(x)$ are all less than 1, so that $[x^n](q(x)/p_k(x)) \rightarrow 0$ with n . Thus we will only be concerned with finding A and B . Multiplying by $(1-x)^2$ and setting $x = 1$ it follows that $A = 1/(1+p_k(1)) = 1/(k+1)$. Since $(1-x)$ is a double root and we wish to avoid considering the roots of $p_k(x)$ explicitly, we will determine B from the partial fraction expansion of $F_k(x) - A/(1-x)^2$. It can be shown that for $k \geq 1$,

$$F_k(x) - \frac{A}{(1-x)^2} = \frac{(k-1) \left(\sum_{i=1}^{k-2} x^i \right) - \left(\sum_{i=1}^{k-2} ix^i \right) - 2}{(k+1)(2-x-x^k)} = \frac{B}{(1-x)} + \frac{q(x)}{p_k(x)}$$

which yields the value of B to be

$$B = \left. \frac{(k-1) \left(\sum_{i=1}^{k-2} x^i \right) - \left(\sum_{i=1}^{k-2} ix^i \right) - 2}{(k+1)p_k(x)} \right|_{z=1} = \frac{(k-1)(k-2) - 4}{2(k+1)^2}. \quad (8)$$

Since $\mathbf{E}[w(e)] = A(n+1) + B + o(1)$, and $\mathbf{E}[m(e)] = 2^{k-1} + \mathbf{E}[w(e)] - 2$, we have proven (3). To determine $\mathbf{Var}[m(e)]$, let $H_k(x) = G''_k(x/2, 1)$ where

$$H_k(x) = \frac{2x^{k+1}}{(1-x)(2-x-x^k)^2} = \frac{2x^{k+1}}{(1-x)^3 p_k(x)^2} \quad (9)$$

k	2	3	4	5	6	7	8	9	10
$ \alpha_k $	0.5	0.707	0.823	0.885	0.922	0.944	0.956	0.969	0.976
C_k	0.296	0.148	0.058	0.0	-0.038	-0.065	-0.084	-0.098	-0.109

Table 1: Tabulation of the constants for Corollary 2.1.

and $p_k(x)$ is as in (6). We note that $\mathbf{Var}[(w(e))] = [x^n](H_k(x) + F_k(x)) - ([x^n]F_k(x))^2$, and $\mathbf{Var}[(m(e))] = \mathbf{Var}[(w(e))]$. The partial fraction expansion of $H_k(x)$ can be shown to have the form

$$H_k(x) = \frac{2}{(k+1)^2(1-x)^3} - \frac{2(1+3k)}{(k+1)^3(1-x)^2} + \frac{r(x)}{(1-x)p_k(x)^2}, \quad (10)$$

which implies that

$$[x^n]H_k(x) = \frac{(n+1)(kn+n-4k)}{(k+1)^3} + O(1). \quad (11)$$

Simplifying the expression for $\mathbf{Var}[(w(e))]$ in terms of $[x^n]H_k(x)$ and $[x^n]F_k(x)$ yields (4). \square

The formulas in the statement of Corollary 2.1 both contain low order terms, which we now consider. The low order term in $\mathbf{E}[m(e)]$ corresponds to $o(1) = O(|\alpha_k|^n)$, where $\alpha_k < 1$ and is the root of largest modulus of $x^k p_k(1/x) = 2x^k - x^{k-1} - 1$, the reflected polynomial of $p_k(x)$. The low order term of $\mathbf{Var}[m(e)]$ corresponds to $O(1) = C_k + O(n|\alpha_k|^n)$ where C_k is $C_k/(1-x)$ in the partial fraction expansion for $H_k(x)$. The value of C_k was not determined in (10) as the tedious algebra required to do so only yields a small increase in the accuracy of (4). Both α_k and C_k are tabulated in Table 1 for k in the range $2 \leq k \leq 10$.

n	k	$\mathbf{E}[m_{nk}(e)]$	$\mathbf{Var}[m_{nk}(e)]$	0.50	0.60	0.75	0.90	0.95	0.99
512	4	108.6	8.3	5	5	6	10	13	29
512	5	99.6	4.8	4	4	5	7	10	23
512	6	103.4	3.1	3	3	4	6	8	18
512	7	126.3	2.1	3	3	3	5	7	15
1024	4	211.0	16.5	6	7	9	13	19	41
1024	5	184.9	9.6	5	5	7	10	14	31
1024	6	176.6	6.1	4	4	5	8	12	25
1024	7	190.3	4.1	3	4	5	7	10	21

Table 2: The distribution of $m_{nk}(e)$ for $n \in \{512, 1024\}$, and k in the range $4 \leq k \leq 7$. The columns show $\delta(m_{nk}(e), p)$, $p \in \{0.50, 0.60, 0.75, 0.90, 0.95, 0.99\}$.

Table 2 shows $\mathbf{E}[m(e)]$ and $\mathbf{Var}[m(e)]$ for several relevant values of n and k , which were calculated exactly by expanding $F_k(x)$ and $H_k(x)$ as power series. The table also gives a measure of the deviation from the mean in terms of the function $\delta(X, p)$ defined as

$$\delta(X, p) = \min \left[\frac{\sigma^2}{d^2} < (1-p) \right], \quad (12)$$

which states that d is the smallest value for which $\Pr(|X - \mu| < d) > p$ according to bounds derived by Chebyshev's inequality. Even using this bound we see that $m(e)$ is concentrated around its expectation, which is thus representative of the distribution as a whole.

Accurate statements concerning $m(e)$ can be made using Table 2. For example when $n = 1024$ we find that $\mathbf{E}[m(e)]$ is minimized when $k = 6$, which is the usual criterion for selecting k for a given n . The $m(e)$ value for $k = 6$ and $k = 7$ are close, but the table shows that for over half the exponents $m(e) < (176.6 + 4) < 181$ when $k = 6$, while $m(e) > (184.9 - 5) > 179$ when $k = 7$, indicating that $k = 6$ is the better choice. Further since Chebyshev's inequality is a general bound, the deviation from $\mathbf{E}[m(e)]$ will be even smaller than as suggested in Table 2, so the separation between $m(e)$ for the cases of $k = 6$ and $k = 7$ will be more pronounced in practice. When $k = 6$ is used, at least three quarters of all exponents will require less than 176.6 ± 5 multiplications, and at least 99% of exponents will require no more than $202 > (176.6 + 25)$ multiplications. On the other hand, almost all exponents will require $151 < (176.6 - 25)$ multiplications.

Of course the optimal choice of k depends not only on $m(e)$ but also on $s(e)$, the number of squarings. However in the next section we show that $\mathbf{E}[s(e)] = n + 1 - k + O(1)$ with an essentially constant variance. Thus the difference in $s(e)$ between various values of k considered in practice (such as $k = 5, 6, 7$ for 1024-bit exponents) is small almost always, so the dominant consideration is to minimize $m(e)$. This statement will need to be reconsidered in groups where there is a significant difference in the cost of a single squaring as compared to a single multiplication. Over the integers, squaring is faster than multiplication by at most a factor of 2 [9, p.597].

Our analysis in this section of $m(e)$ is based on its relation to the number of windows $w(e)$ produced by the sliding window partition, given as $m(e) = 2^{k-1} + w(e) - 2$. We then modeled the distribution of $w(e)$ via a regular expression, but other recurrences could have been used for this purpose. For example, Hui and Lam [6] defined $\mathbf{E}[w(e)] = f_{n,k}$, and then stated that

$$f_{n,k} = \begin{cases} \frac{2^n - 1}{2^n} & \text{if } 0 \leq n \leq k, \\ \frac{1}{2} + \frac{f_{n-1,k}}{2} + \frac{f_{n-k,k}}{2} & \text{if } k < n. \end{cases} \quad (13)$$

The $k < n$ case follows from observing that if the leading bit of e is one then an additional $1 + f_{n-k,k}$ windows will be required, and if the leading bit is zero then an additional $f_{n-1,k}$ windows will be required. We now solve this recurrence and show equivalence to $G_k(x, z)$.

Lemma 2.1 Let $F_k(x) = \sum_{n \geq 0} f_{n,k} x^n$ be the generating function for the sequence $f_{n,k}$. Then

$$F_k(x) = \frac{x}{(1-x)(2-x-x^k)}. \quad (14)$$

Proof. Assuming that $f_{n,k} = 0$ for $n < 0$ then for all n , (13) can be written as

$$2f_{n,k} = 1 + f_{n-1,k} + f_{n-k,k} - [n \leq 0], \quad (15)$$

where $[n \leq 0]$ is a boolean predicate evaluating to 0 or 1. Then summing on x^n it follows that

$$2F_k(x) = \sum_n x^n + \sum_n f_{n-1,k} x^n + \sum_n f_{n-k,k} x^n - \sum_n [n \leq 0]. \quad (16)$$

But since $\sum_n x^n - \sum_n [n \leq 0] = x/(1-x)$ then

$$2F_k(x) = xF_k(x) + x^k F_k(x) + \frac{x}{1-x} \quad (17)$$

from which the lemma follows. \square

It is easy to verify that

$$G'_k(x/2, 1) = \left. \frac{\partial G_k(x/2, z)}{\partial z} \right|_{z=1} = F_k(x). \quad (18)$$

Similar recurrences to $F_k(x)$ were also derived by Cohen [3], however we argue that the generating function $G_k(x, z)$ is preferable over both alternatives since the variance of $m(e)$ is available directly from $G_k(x, z)$

3 The squaring distribution

To analyse the squaring distribution it is convenient to consider an exponent e being partitioned into $w(e)$ windows $W_1, W_2, \dots, W_{w(e)}$ of the form

$$e = 0^{j_0} W_1 0^{j_1} W_2 0^{j_2} \dots 0^{j_{w(e)-1}} W_{w(e)} 0^{j_{w(e)}} \quad (19)$$

where $j_i \geq 0$ for $0 \leq i \leq w(e)$ (assuming the convention that $0^0 = \epsilon$, the empty string), and $W_i = 1$ or $W_i = 1(1+0)^{d_i}1$ with $0 \leq d_i \leq k-2$ for $1 \leq i \leq w(e)$. From (19) and the precomputation, the number of required squarings is $s(e) = 1 + \sum_{i=1}^{w(e)} (j_i + |W_i|) \stackrel{\text{def}}{=} 1 + (n - \beta(e))$ where $\beta(e) = j_0 + |W_1|$. In the next theorem we use regular languages to determine the distribution of $\beta(e)$, and hence the distribution of $s(e)$.

Theorem 3.1 Let $s(e) = s_{nk}(e)$ be the number of squarings required by the sliding window method to compute g^e for the n -bit exponent e using k -bit windows. Then for a uniformly distributed n -bit exponent e

$$\begin{aligned} \mathbf{E}[s_{nk}(e)] &= n + 1 - k - 2^{-k} + O(n/2^n), \\ \mathbf{Var}[s_{nk}(e)] &= 4 + \frac{1-2k}{2^{k-1}} - \frac{4}{2^{2k}} + O(n^2/2^n). \end{aligned}$$

Proof. The theorem can be proved by determining the distribution of $\beta(e)$. To this end, consider the following regular expression $R = R_1 + R_2 + R_3$ where

$$R = 0^* + 0^* \left(\sum_{i=0}^{k-2} 1(1+0)^i \right) + 0^* \left(10^{k-1} + \sum_{i=0}^{k-2} 1(0+1)^{k-2-i} 10^i \right) (1+0)^*.$$

Here R_1 denotes the all-zero string, implying $w(e) = 0$ and $\beta(e) = n$; R_2 denotes strings that end with a window of length less than k , implying $w(e) = 1$, $|W_1| < k$, $j_1 = 0$ and $\beta(e) = n$; finally R_3 denotes strings that have at least one window W_i for which $(|W_i| + j_i) \geq k$, implying $w(e) \geq 1$ and $0 < \beta(e) \leq n$. Clearly all binary strings will be generated by either R_1, R_2 or R_3 , and since these regular expressions are unambiguous, we may use the same enumeration techniques as employed in Theorem 2.1.

As before we will use x to mark length, but z will be used to accumulate $\beta(e)$. If no window exists the expression will be marked using $z^0 = 1$. Then R_1 is marked to give $G_{R_1}(x, z) = 1/(1-x)$, and R_2 is marked as

$$G_{R_2}(x, z) = \frac{xz}{1-xz} \left(\frac{1 - (2xz)^{k-1}}{1 - 2xz} \right). \quad (20)$$

Finally the summation term of R_3 is marked as $\sum_{i=0}^{k-2} (xz)(2xz)^{k-2-i}(xz)x^i$ yielding

$$G_{R_3}(x, z) = \frac{1}{1-xz} \left(zx^k + \frac{x^k}{4} \left(\frac{1-(2z)^{k+1}}{1-2z} - 2z - 1 \right) \right) \frac{1}{1-2x}. \quad (21)$$

Letting $G_k(x, z) = \prod_{i=1}^3 G_{R_i}(x, z)$ we first observe that $G_k(x, 1) = 1/(1-2x)$ as expected. Then it can be shown that for $k \geq 2$,

$$\begin{aligned} G'_k(x/2, 1) &= \frac{2^{k-1} \left(1 + x + \sum_{i=0}^{k-2} x^i \right) - x^{k-1} \left(x + 2^{k-1} - 2 \right)}{2^{k-1}(1-x)(2-x)^2}, \\ &= \frac{k + 2^{1-k}}{(1-x)} + \frac{r_1(x)}{(2-x)^2}, \end{aligned} \quad (22)$$

implying $[x^n]G'_R(x/2, 1) = \mathbf{E}[\beta_{nk}(e)] = (k + 2^{1-k}) + O(n/2^n)$ since $[x^n](2-x)^{-2} = O(n/2^n)$. Similarly for $k \geq 2$, using partial fractions it can be shown that

$$G''_k(x/2, 1) = \frac{4 + k(k-1)}{(1-x)} + \frac{r_2(x)}{(2-x)^3}, \quad (23)$$

implying that $[x^n]G''_k(x/2, 1) = 4 + k(k-1) + O(n^2/2^n)$ since $[x^n](2-x)^{-3} = O(n^2/2^n)$. The theorem now follows by expanding $\mathbf{Var}[s_{nk}(e)]$ in terms of $[x^n]G'_k(x/2, 1)$ and $[x^n]G''_k(x/2, 1)$. \square

Since the variance is essentially constant we find that $s(e)$ is concentrated around its expectation, even more so than $m(e)$.

References

- [1] J. Bos and M. Coster. Addition chain heuristics. *Advances in Cryptology, CRYPTO 89, Lecture Notes in Computer Science, vol. 218, G. Brassard ed., Springer-Verlag*, pages 400–407, 1990.
- [2] G Cohen, A. Lobstein, D. Naccache, and G. Zémor. How to improve an exponentiation black-box. *Advances in Cryptology, EUROCRYPT 98, Lecture Notes in Computer Science, vol. 1403, K. Nyberg, ed., Springer-Verlag*, pages 211–220, 1998.
- [3] H. Cohen. Analysis of the flexible powering algorithm. submitted for publication, available at <http://www.math.u-bordeaux.fr/~cohen/>.
- [4] J. F. Dhem. *Design of an efficient public key cryptographic library for RISC-based smart cards*. PhD thesis, Université catholique de Louvain, 1998. Available at <http://www.dice.ucl.ac.be/crypto/dhem/dhem.html>.
- [5] R. L. Graham, D. E. Knuth, and O. Patshnik. *Concrete Mathematics, A Foundation for Computer Science, First Edition*. Addison Wesley, 1989.
- [6] L. Hui and K.-Y. Lam. Fast square-and-multiply exponentiation for RSA. *Electronics Letters*, 30(17):1396–1397, 1994.
- [7] C. K. Koc. Analysis of sliding window techniques for exponentiation. *Computers and Mathematics with Applications*, 30(10):17–24, 1995.

- [8] D. E. Knuth. *The Art of Computer Programming : Volume 2, Seminumerical Algorithms (3rd Edition)*. Addison Wesley, 1997.
- [9] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996.
- [10] L. J. O'Connor. An analysis of exponentiation based on formal languages. *Advances in Cryptology, EUROCRYPT 99, Lecture Notes in Computer Science, vol. 1592, J. Stern, ed., Springer-Verlag*, pages 375–388, 1999.
- [11] R. Sedgewick and P. Flajolet. *An introduction to the analysis of algorithms*. Addison-Wesley Publishing Company, 1996.
- [12] C. D. Walter. Exponentiation using division chains. *IEEE Transactions on Computers*, 47(7):757–765, 1998.
- [13] M. Wiener. Personal communication.