An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm

Mathieu Ciet[1] and Francesco Sica[2,⋆]

[1] Gemplus Security Technologies Department,
La Vigie, ZI Athélia IV, Av. du Jujubier,
B.P. 100, 13705 La Ciotat Cedex, France
mathieu.ciet@gemplus.com
[2] Mount Allison University – AceCrypt,
Department of Mathematics and Computer Science,
67 York Street, Sackville, NB, E4L 1E6, Canada
fsica@mta.ca – http://www.acecrypt.com

Abstract. In this paper we produce a practical and efficient algorithm to find a decomposition of type

$$n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \quad \text{with} \quad k \leq \big(c + o(1)\big) \frac{\log n}{\log \log n} \ .$$

It is conjectured that one can take $c = 2$ above. Then this decomposition is refined into an effective scalar multiplication algorithm to compute nP on some supersingular elliptic curves of characteristic 3 with running time bounded by

$$O\left(\frac{\log n}{\log \log n}\right)$$

and essentially no storage. To our knowledge, this is the first instance of a scalar multiplication algorithm that requires $o(\log n)$ curve operations on an elliptic curve over $\mathbb{F}_q$ with $\log q \approx \log n$ and uses comparable storage as in the standard double-and-add algorithm.

This leads to an efficient algorithm very useful for cryptographic protocols based on supersingular curves. This is for example the case of the well-studied (in the past four years) identity based schemes. The method carries over to any supersingular curve of fixed characteristic.

Keywords: Integer decomposition, exponentiation algorithms.

1 Introduction

In asymmetric cryptographic algorithms, the costliest part in execution time is most often the computation of nP, for P belonging to some group[1] denoted

⋆ This work was partially supported by a NSERC Discovery Grant

[1] This operation is called exponentiation or scalar multiplication, according to the multiplicative, respectively additive, notation of the group law.

E. Dawson and S. Vaudenay (Eds.): Mycrypt 2005, LNCS 3715, pp. 171–182, 2005.

additively. The standard algorithm to perform this is to decompose n in base 2 and to apply a double-and-add algorithm. If n is randomly chosen with a fixed number of bits, then this algorithm requires on average $\log_2 n$ doublings and $(\log_2 n)/2$ additions. Hence we can say that a classical double-and-add algorithm takes time[2] $\sim c \log n$ (asymptotic to $c \log n$ on average) for some $c > 0$.

In some specific groups, one can improve the constant c by taking for instance signed binary expansions [10] or expansions to other bases [3,11]. We call *linear* an algorithm with running time T satisfying $c \log n < T < d \log n$ for some $c, d > 0$. Similarly, we shall call it *sublinear* if the running time is $o(\log n)$ (little oh) as n goes to infinity. This means that this time is strictly smaller than $\epsilon \log n$ for any $\epsilon > 0$.

The aim of the present work is to present the first practical sublinear scalar multiplication algorithm. The algorithm is sublinear when used on particular supersingular elliptic curves of fixed characteristic (for instance, elliptic curves defined over $\mathbb{F}_p$). We shall deal with characteristic 3 here as an illustration of the method and leave the easy modifications to the interested reader.

The algorithm proceeds as follows. We first decompose the multiplier n as

$$n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \tag{1}$$

with $(s_i, t_i) \neq (s_j, t_j)$ for $i \neq j$ and

$$k \leq \big(c + o(1)\big) \frac{\log n}{\log \log n}$$

for some proven $c > 0$ (conjecturally $c = 2$). This allows us to build a double-and-add algorithm where the number of additions is $O(\frac{\log n}{\log \log n})$.

Then a simple remark allows us to impose a further condition that $\max(s_i) \leq \log^{1-\delta} n$ for any $0 < \delta < 1$. This has the effect of bringing the number of doublings down to $O(\log^{1-\delta} n)$.

We still need to worry about triplings, but on a supersingular curve these are practically as fast a two Frobenius endomorphisms, hence they can be neglected if normal bases are used and even in the case of polynomial bases, they are faster than a doubling or an addition.

Therefore the total running time of our algorithm will be bounded by the number of additions, that is $O(\frac{\log n}{\log \log n})$.

Let us mention that the idea of using (1), called in the literature double base number system, goes back at least to [5], where even an exponentiation algorithm is given [6] with comparable running time as ours. However this algorithm requires $O(\log^2 n)$ in storage, whereas our algorithm only needs a minimal storage. Also [4] develops these ideas into a generic scalar multiplication algorithm that would have fast running time. Unfortunately, no explicit estimate is given at the moment.

[2] The time unit is an elliptic curve operation, say an addition, since a doubling takes a positive proportion of performing an addition.