Extended Double-Base Number System with Applications to Elliptic Curve Cryptography

Christophe Doche[1] and Laurent Imbert[2]

[1] Department of Computing
Macquarie University, Australia
doche@ics.mq.edu.au
[2] LIRMM, CNRS, Université Montpellier 2, UMR 5506, France
& ATIPS, CISaC, University of Calgary, Canada
Laurent.Imbert@lirmm.fr

Abstract. We investigate the impact of larger digit sets on the length of Double-Base Number system (DBNS) expansions. We present a new representation system called *extended DBNS* whose expansions can be extremely sparse. When compared with double-base chains, the average length of extended DBNS expansions of integers of size in the range 200–500 bits is approximately reduced by 20% using one precomputed point, 30% using two, and 38% using four. We also discuss a new approach to approximate an integer n by $d2^a3^b$ where d belongs to a given digit set. This method, which requires some precomputations as well, leads to realistic DBNS implementations. Finally, a left-to-right scalar multiplication relying on extended DBNS is given. On an elliptic curve where operations are performed in Jacobian coordinates, improvements of up to 13% overall can be expected with this approach when compared to window NAF methods using the same number of precomputed points. In this context, it is therefore the fastest method known to date to compute a scalar multiplication on a generic elliptic curve.

Keywords: Double-base number system, Elliptic curve cryptography.

1 Introduction

Curve-based cryptography, especially elliptic curve cryptography, has attracted more and more attention since its introduction about twenty years ago [1,2,3], as reflected by the abundant literature on the subject [4,5,6,7]. In curve-based cryptosystems, the core operation that needs to be optimized as much as possible is a scalar multiplication. The standard method, based on ideas well known already more than two thousand years ago, to efficiently compute such a multiplication is the double-and-add method, whose complexity is linear in terms of the size of the input. Several ideas have been introduced to improve this method; see [8] for an overview. In the remainder, we will mainly focus on two approaches:

- Use a representation such that the expansion of the scalar multiple is sparse. For instance, the non-adjacent form (NAF) [9] has a nonzero digit density of

R. Barua and T. Lange (Eds.): INDOCRYPT 2006, LNCS 4329, pp. 335–348, 2006.

1/3 whereas the average density of a binary expansion is 1/2. This improvement is mainly obtained by adding -1 to the set $\{0, 1\}$ of possible coefficients used in binary notation. Another example is the double-base number system (DBNS) [10], in which an integer is represented as a sum of products of powers of 2 and 3. Such expansions can be extremely sparse, *cf.* Section 2.
- Introduce precomputations to enlarge the set of possible coefficients in the expansion and reduce its density. The k-ary and sliding window methods as well as window NAF methods [11,12] fall under this category.

In the present work, we mix these two ideas. Namely, we investigate how precomputations can be used with the DBNS and we evaluate their impact on the overall complexity of a scalar multiplication.

Also, computing a sparse DBNS expansion can be very time-consuming although it is often neglected when compared with other representations. We introduce several improvements that considerably speed up the computation of a DBNS expansion, *cf.* Section 4.

The plan of the paper is as follows. In Section 2, we recall the definition and basic properties of the DBNS. In Section 3, we describe how precomputations can be efficiently used with the DBNS. Section 4 is devoted to implementation aspects and explains how to quickly compute DBNS expansions. In Section 5, we present a series of tests and comparisons with existing methods before concluding in Section 6.

2 Overview of the DBNS

In the *Double-Base Number System*, first considered by Dimitrov *et al.* in a cryptographic context in [13], any positive integer n is represented as

$$n = \sum_{i=1}^{\ell} d_i 2^{a_i} 3^{b_i}, \text{ with } d_i \in \{-1, 1\}. \tag{1}$$

This representation is obviously not unique and is in fact highly redundant. Given an integer n, it is straightforward to find a DBNS expansion using a greedy-type approach. Indeed, starting with $t = n$, the main task at each step is to find the $\{2, 3\}$-integer z that is the closest to t (i.e. the integer z of the form $2^a 3^b$ such that $|t - z|$ is minimal) and then set $t = t - z$. This is repeated until t becomes 0. See Example 2 for an illustration.

Remark 1. *Finding the best $\{2, 3\}$-approximation of an integer t in the most efficient way is an interesting problem on its own. One option is to scan all the points with integer coordinates near the line $y = -x \log_3 2 + \log_3 t$ and keep only the best approximation. A much more sophisticated method involves continued fractions and Ostrowski's number system, cf. [14]. It is to be noted that these methods are quite time-consuming. See Section 4 for a more efficient approach.*