The Double-Base Number System in Elliptic Curve Cryptography

Christophe Doche
Department of Computing
Division of Information and Communication Sciences
Macquarie University
North Ryde, NSW 2109
Australia

Laurent Imbert
PIMS, CNRS & University of Calgary
Centre for Information Security and Cryptography
2500 University Dr. NW
Calgary, T2N 1N4
Canada

***Abstract*— Three algorithms for double-scalar multiplication on elliptic curves, based on the representations of a pair of integers as mixed powers of 2 and 3, are presented.**

I. Introduction

Elliptic curves have been known for several centuries and have been used to tackle Diophantine problems, e.g. Fermat last theorem. One fundamental property of elliptic curves, is that it is possible to add 2 points lying on a curve, and this addition is in fact a group law (see e.g. [1] for a good introduction). For instance, given a point P and an integer k, it is possible to build the point $Q = kP$ that is also on the curve. This operation is called a scalar multiplication.

More recently, elliptic curves have been used for public-key cryptography purposes [2], [3]. The security of elliptic curve-based cryptographic protocols, such as signature or encryption, is based on the so-called discrete logarithm problem, which, in some sense, is the converse of the scalar multiplication: Given a point Q that is a certain multiple k of a fixed point P, can we find the value of k (in a reasonably amount of time)? The only known algorithms to solve this problem (for a well-chosen elliptic curve) all have exponential-time complexity. Note that other one-way problems commonly used in public-key cryptography, i.e. factorization or discrete logarithm problem in a finite field, can be solved with sub-exponential time algorithms.

This explains the recent attention received by elliptic curves. But the difficulty of the discrete logarithm problem can only be exploited if scalar multiplications are easy to obtain. Fortunately, a scalar multiplication can always be computed in linear time, nonetheless this operation needs to be optimized as much as possible. The methods to compute a scalar multiplication are in fact linked to the different representations of the scalar k. For instance, k is often represented in base 2, or in base 2^w. Also, it has been suggested to use signed-digit representations where the coefficients in the expansion are allowed to be negative. See [4, chapter 3] for a survey on classical point multiplication algorithms.

More recently, the concept of double-base number system (DBNS) has shown some advantages in implementing elliptic-curve scalar multiplication [5], [6], [7], [8]. In DBNS, an integer n is represented as a sum (or difference) of mixed powers of 2 and 3:

$$n = \sum_{i=0}^{m} \pm 2^{a_i} 3^{b_i}.$$

Such a representation can be found easily with a greedy algorithm and the number of terms in the sum is proved to be sublinear. More precisely, it is in $O(\log n / \log\log n)$. This DBNS has been successfully used for a certain category of curves, i.e. supersingular curves. For ordinary curves, or more generally for curves not admitting a free endomorphism, a variation of the DBNS, called Double-Base chains has to be considered. Several algorithms can be used to find such DB-chains, including a variant of the greedy algorithm for DBNS.

Some applications, like ECDSA signature verification, need fast computation of double-scalar multiplication; i.e. $kP + lQ$ for $k, l > 0$ and P, Q two points on the curve. In this paper, we present three approaches to this problem, based on joint DB-chains.

II. Double-base chains

A double-base chain for k is an expansion of the form

$$k = \sum_{i=1}^{n} d_i 2^{a_i} 3^{b_i}, \tag{1}$$

with $d_i \in \{-1, 1\}$ and such that the exponents (a_i, b_i) decrease for the product order.

For example, a double-base chain computing 1717 is given by $1717 = 2^6 3^3 - 2^2 3^1 + 2^0 3^0 = 1728 - 12 + 1$. In order to compute $1717P$, one uses a Horner-like algorithm by considering the differences between consecutive pairs of exponents and by applying doublings and triplings and additions/subtractions accordingly.

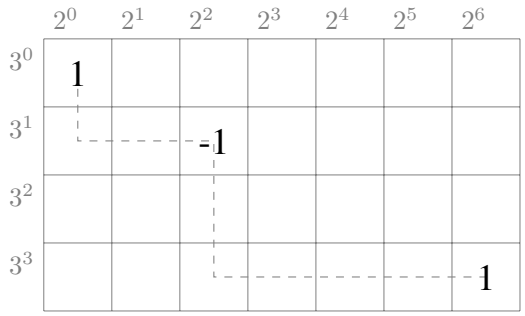

Fig. 1. An example of staircase walk for a double-base chain representing 1717

For example, starting from P, one successively computes

$$\begin{aligned} 2^{6-2}P &= 16P \\ 3^{3-1}(16P) &= 144P \\ 144P - P &= 143P \\ 2^{2-0}(143P) &= 572P \\ 3^{1-0}(572P) &= 1716P \\ 1716P + P &= 1717P \end{aligned}$$

using a total of 6 doublings, 3 triplings and 3 additions. Note that this cost is given by the powers of the largest term and by the number of summands in (1).

An easy way to visualize expansions using two bases (say e.g. 2 and 3), is to use a two-dimensional array (the columns represent the powers of 2 and the rows represent the powers of 3) into which each non-zero cell contains the sign of the corresponding term. (by convention, the upper-left corner corresponds to $2^0 3^0 = 1$.) A double-base chain can thus be represented by a staircase walk from the bottom-right corner to the top-left corner, with non-zero digits distributed along this path. An example of such a double-base chain is shown in Fig. 1. (Since a given set of non-zero cells can lead to many different staircase walks, we adopt the convention to walk West as much as we can before going North.)

In the next section, we consider the problem of computing joint double-base chains for a pair of integers. We propose several algorithms which compute two double-base chains that share the same staircase walk; only the distribution of the digits along the path differ.

III. Double scalar multiplication algorithms

A signature verification essentially requires a double-scalar multiplication; i.e. a computation of the form $kP+lQ$ for two points P, Q and two positive integers k, l. Obviously, kP and lQ can be computed independently and added together at the cost of $|k|+|l|-2$ doublings and $\frac{|k|+|l|}{2}$ additions on average (where $|x|$ denotes the binary length of x).

More interestingly, $kP + lQ$ can also be obtained as the result of a combined operation. So-called Shamir's trick can be used to represent joint expansions using a $2 \times t$ matrix

$$\begin{array}{rl} k = & (k_{t-1} \quad \dots \quad k_1 \quad k_0) \\ l = & (l_{t-1} \quad \dots \quad l_1 \quad l_0), \end{array}$$

with $k_i, l_i \in \{-1, 0, 1\}$ for all i. The number of additions required by Shamir's simultaneous method is equal to the so-called joint Hamming weight; i.e., the number of non-zero columns. For example, if k and l are both written in the Non-Adjacent Form [9], [10], the computation of $kP + lQ$ costs $t + 1$ doublings and $5t/9$ additions on average.

Example 1: The 2×9 matrix given by the NAFs of $k = 145$ and $l = 207$

$$\begin{array}{rl} 145 = & (0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1) \\ 207 = & (1 \quad 0 \quad \bar{1} \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad \bar{1}) \end{array}$$

has joint Hamming weight 5.

In [11], Solinas introduced the Joint Sparse Form (JSF) to further reduce the average number of non-zero columns. The main idea behind Solinas' algorithm is to make sure that out of three consecutive columns, at least one is a zero-column. Solinas' algorithm is given in terms of arithmetic operations but it basically reduces to computations modulo 8 (bit operations). By carefully choosing the positive/negative values of the remainders (mod 8), Solinas proves the uniqueness and optimality (in the context of joint signed binary expansions) of the JSF, showing that the computation of $kP+lQ$ requires t doublings and $t/2$ additions on average.

Example 2: Using the same values as above ($k = 145$, $l = 207$), the JSF

$$\begin{array}{rl} 145 = & (1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1) \\ 207 = & (1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad \bar{1}) \end{array}$$

has Hamming weight 4.

The simultaneous methods described above require precomputations of points involving both P and Q. For example, the JSF algorithm needs the points $P + Q$ and $P - Q$ to be precomputed.

On the other hand, another class of algorithms, called interleaving methods, use precomputed values that only involve a single point, which allows the use of different methods for each scalar (such as different width-w NAFs); the doubling steps being done jointly. When the same width-w NAF is used for both k and l, the overall cost of interleaving methods is t doublings and $2t/(w+1)$ additions on average (see [4, pp 111–113] for more details).

IV. Double-base chains for a pair of integers

A. Hybrid binary-ternary joint sparse form

In Algorithm 1, the hybrid binary-ternary joint form of a pair of integers is calculated by first checking whether both k and l are divisible by 3. If this is the case, both digits are set to 0 and the base set to 3, otherwise we check whether they are both divisible by 2 and proceed accordingly. Finally, if the pair is not divisible by 3 or 2, we make both numbers divisible by 6 by subtracting k_i mods $6 \in \{-2, -1, 0, 1, 2, 3\}$ from k_i, and then divide the results by 2. Therefore, in the next step, both numbers are divisible by 3 and we generate a zero column. This approach is an extension of the binary/ternary method proposed by Ciet et al. in [12].

Algorithm 1 Hybrid binary-ternary joint form (HBTJF)

INPUT: Two positive integers k, l
OUTPUT: Arrays `hbtk[], hbtl[], base[]`

$i = 0$
while $k > 0$ or $l > 0$ do
  if $k \equiv 0 \pmod 3$ and $l \equiv 0 \pmod 3$ then
    `base[`i`]` = 3;
    `hbtk[`i`]` = `hbtl[`i`]` = 0;
    $k = k/3$; $l = l/3$;
  else if $k \equiv 0 \pmod 2$ and $l \equiv 0 \pmod 2$ then
    `base[`i`]` = 2;
    `hbtk[`i`]` = `hbtl[`i`]` = 0;
    $k = k/2$; $l = l/2$;
  else
    `base[`i`]` = 2;
    `hbtk[`i`]` = k mods 6; `hbtl[`i`]` = l mods 6;
    $k = (k - $ `hbtk[`i`]`$)/2$; $l = (l - $ `hbtl[`i`]`$)/2$;
  $i = i + 1$
return `hbtk[], hbtl[], base[]`

TABLE I
PRECOMPUTATIONS FOR HBTJF SCALAR MULTIPLICATION

	P	-	-
Q	$P \pm Q$	$2P \pm Q$	$3P \pm Q$
-	$P \pm 2Q$	-	$3P \pm 2Q$
-	$P \pm 3Q$	$2P \pm 3Q$	-

Example 3: For $k = 1225$ and $l = 723$ Algorithm 1 returns

$$\begin{aligned} 1225 &= (3 \quad 0 \quad \bar{1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 1) \\ 723 &= (2 \quad 0 \quad \bar{2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 3) \\ \texttt{base[]} &= (2 \quad 3 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 2) \end{aligned}$$

only requires 8 digits and has joint Hamming weight 3. Note that for the same pair on integers, the joint Hamming weight is equal to 7 and the interleaving w-NAF (with $w = 5$ for 1225 and $w = 4$ for 723) has 6 non-zero elements[1]

Since the HBTJF uses the digit set $\{-2, -1, 0, 1, 2, 3\}$, a total of 14 points have to be precomputed (see Table I). Note that the points $2P, 2Q, 3P, 3Q$ are not needed as they correspond to pairs of integers that are simultaneously divisible by 2 or 3. Also, since the negation of a point is negligible, only one set of point difference need to be calculated; for example, $2Q - P$ is easily deduced from $P - 2Q$.

The behavior of Algorithm 1 can be theoretically analyzed. Using probabilistic tools, it can be shown [13] that the average number of elliptic curve additions per bit is approximately

$$\frac{24}{59} \times 0.7888 \approx 0.3209.$$

Assuming $k \geq l$, the number of doublings and triplings are approximately equal to $0.43 \log_2 k$ and $0.36 \log_2 k$ respectively.

[1]In the interleaving method, the non-zero elements in both w-NAF representations are considered, instead of joint Hamming weight.

B. Greedy approach

One disadvantage of the hybrid approach is the relatively high number of precomputations. A joint double-base chain; i.e. an expansion of the form

$$\binom{k}{l} = \sum_{i=1}^{n} \binom{c_i}{d_i} 2^{a_i} 3^{b_i}, \tag{2}$$

with $(c_i, d_i) \in \{-1, 0, 1\}$ and (a_i, b_i) decrease for the product order, can be computed using the following extension of the greedy method.

At each step, given the pair of positive integers (x, y), choose an approximation (x', y') among $(2^a 3^b, 0)$, $(0, 2^a 3^b)$, $(2^a 3^b, 2^a 3^b)$ such that the quantity $(x - x')^2 + (y - y')^2$ is minimal. Accordingly, the coefficients in the double-base chain will be $\binom{\pm 1}{0}$, $\binom{0}{\pm 1}$, $\binom{\pm 1}{\pm 1}$.

Thus this method relies on only two precomputations just like for the Joint Sparse Form. However, it is difficult to analyze. The next method appears to be more efficient in practice and is also straightforward to analyze.

C. Joint Binary-Ternary

Given a pair (k, l) of positive integers, let $v_p(k, l)$ denote the largest power of p that divides k and l simultaneously. Then proceeds as follows:

1) divide k and l by $2^{v_2(k,l)} 3^{v_3(k,l)}$ such that the result (k', l') satisfy $v_2(k', l') = v_3(k', l') = 0$,
2) from all the pairs of the form $(k' + c, l' + d)$ for $c, d \in \{-1, 0, 1\}$, select the one such that $2^{v_2(k'+c,l'+d)} 3^{v_3(k'+c,l'+d)}$ is maximal. (If several pairs achieve the same maximum, any pair can be chosen),
3) divide the selected pair $(k' + c, l' + d)$ by 2 and by 3 as much as possible and iterate the process as long as $k > 1$ or $l > 1$.

A probabilistic approach can be used to prove that the average number of addition per bit associated to an expansion of the form of (2) returned by this algorithm belongs to the interval $[0.3942, 0.3945]$. Also, if $k \geq l$, the number of doublings and triplings are approximately equal to $0.55 \log_2 k$ and $0.28 \log_2 k$ respectively.

Compared to the (binary) Joint Sparse Form, this tree-based approach requires the same precomputations, namely $P + Q$ and $P - Q$, whereas the number of additions is reduced by 21% on average. Finally, note that this method readily generalizes to larger digit sets, simply by considering more pairs in step 2 of the above description of the algorithm; i.e., by allowing c, d to belong to larger digits sets. This can be combined with a tree-based search for even better results. See [14] for details.

V. CONCLUSIONS

Elliptic curve digital signature verifications requires a double-scalar multiplication; i.e. the computation of an expression of the form $kP + lQ$ for some points P, Q lying on an elliptic curve, and some positive integers k, l. We have introduced three algorithms based on joint double-base chains. These approaches uses triplings and are therefore of potential

interest for curves for which this operation can be computed efficiently.

REFERENCES

[1] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2003.

[2] V. S. Miller, "Uses of elliptic curves in cryptography," in *Advances in Cryptology, CRYPTO'85*, ser. Lecture Notes in Computer Science, vol. 218. Springer, 1986, pp. 417–428.

[3] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan. 1987.

[4] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[5] V. Dimitrov, L. Imbert, and P. K. Mishra, "The double-base number system and its application to elliptic curve cryptography," *Mathematics of Computation*, vol. 77, no. 262, pp. 1075–1104, 2008.

[6] C. Doche and L. Imbert, "Extended double-base number system with applications to elliptic curve cryptography," in *Progress in Cryptology, INDOCRYPT'06*, ser. Lecture Notes in Computer Science, vol. 4329. Springer, 2006, pp. 335–348.

[7] R. Avanzi, V. Dimitrov, C. Doche, and F. Sica, "Extending scalar multiplication using double bases," in *Advances in Cryptology, ASIACRYPT'06*, ser. Lecture Notes in Computer Science, vol. 4284. Springer, 2006, pp. 130–144.

[8] V. S. Dimitrov, K. Järvinen, M. J. Jacobson, Jr., W. F. Chan, and Z. Huang, "FPGA implementation of point multiplication on Koblitz curves using Kleinian integers," in *Cryptographic Hardware and Embedded Systems, CHES'06*, ser. Lecture Notes in Computer Science, vol. 4249. Springer, 2006, pp. 445–459.

[9] G. W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, vol. 1, pp. 231–308, 1960.

[10] M. Joye and S.-M. Yen, "Optimal left-to-right binary signed-digit exponent recoding," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 740–748, 2000.

[11] J. A. Solinas, "Low-weight binary representations for pairs of integers," Center for Applied Cryptographic Research, University of Waterloo, Waterloo, ON, Canada, Research report CORR 2001-41, 2001.

[12] M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery, "Trading inversions for multiplications in elliptic curve cryptography," *Designs, Codes and Cryptography*, vol. 39, no. 2, pp. 189–206, May 2006.

[13] J. Adikari, V. Dimitrov, and L. Imbert, "Hybrid binary-ternary joint sparse form and its application in elliptic curve cryptography," Cryptology ePrint Archive, Report 2008/285, 2008, http://eprint.iacr.org/ (Submitted).

[14] C. Doche, D. Kohel, and F. Sica, "Double-base number system for multi-scalar multiplications," Cryptology ePrint Archive, Report 2008/388, 2008, http://eprint.iacr.org/.