Hybrid Binary-Ternary Number System for Elliptic Curve Cryptosystems

Jithra Adikari, *Student Member*, *IEEE*, Vassil S. Dimitrov, and Laurent Imbert

Abstract—Single and double scalar multiplications are the most computational intensive operations in elliptic curve based cryptosystems. Improving the performance of these operations is generally achieved by means of integer recoding techniques, which aim at minimizing the scalars' density of nonzero digits. The hybrid binary-ternary number system provides both short representations and small density. In this paper, we present three novel algorithms for both single and double scalar multiplication. We present a detailed theoretical analysis, together with timings and fair comparisons over both tripling-oriented Doche-Ichart-Kohel curves and generic Weierstrass curves. Our experiments show that our algorithms are almost always faster than their widely used counterparts.

Index Terms—Elliptic curve cryptography, single/double scalar multiplication, hybrid binary-ternary number system, DIK-3 curves.

1 Introduction

Even though elliptic curves have been studied for more than a 100 years, their practical use for public key cryptography has only been proposed at the end of the 1980s independently by Koblitz [1] and Miller [2]. Since then elliptic curve cryptography (ECC) has drawn lots of attention from different research communities [3], [4], [5]. The probable intractability of the elliptic curve discrete logarithm problem (ECDLP) represents a major advantage of elliptic curves over other problems used for public key cryptography such as the discrete logarithm problem in the multiplicative group $\mathbf{Z}_p^*$, used for Diffie-Hellman key exchange [6], or the famous integer factorization problem used for RSA [7], since it leads to shorter key lengths.

In ECC-based cryptographic protocols, most of the computational power is dedicated to single and multiple-point multiplication. Let E be an elliptic curve defined over a field $\mathbf{K}$, P a point of order n in the group $E(\mathbf{K})$, and k an integer. The computation which consists in adding P to itself $k-1$ times, denoted $[k]P = P + \cdots + P$, is called single scalar multiplication. If we consider a second point Q in $E(\mathbf{K})$ and another integer l, the computation of $[k]P + [l]Q$ is called double-point multiplication.

In the single scalar case, the binary representation of a t-bit scalar k has $t/2$ nonzero bits on average, leading to $t-1$ point doublings and $t/2$ point additions on average. Since the cost of point negation is negligible over elliptic curves, we can use signed binary representations [8], [9], [10], [11] with digits $\{-1, 0, 1\}$. A generalized concept of signed binary representation called window nonadjacent form (w-NAF) can be used to further reduce the number of nonzero elements [12]. Dimitrov et al. have recently proposed an efficient algorithm for computing $[k]P$ using double-base chains [13], [14]. This idea has been extended to windowed double-base chains in [15] by Doche and Imbert. More discussions and methods based on chain representations are available in [16], [17], [18], [19]. Very recently, Méloni and Hasan proposed to combine double-base representations and Yao's algorithm [20]. Extending the double-base concept further, number of multibase variants have been proposed for single scalar multiplication[1] (see [21], [22]). At the end of this paper, we present many implementation results and comparisons based on standardized curves and parameters.

In the double scalar multiplication, we need to perform $t-1$ point doublings plus $3t/4$ point additions on average (assuming two t-bit scalars). In [23], Solinas proposed an algorithm to compute a minimal joint representation for two scalars. The new representation, called joint sparse form (JSF), requires only $t/2$ point additions and only two precomputations, namely the points $P+Q$ and $P-Q$. Later, Hankerson et al. introduced the interleaving w-NAF method [12]. The number of point additions involved in this algorithm is as low as $2t/(w+1)$ for two t-bit numbers while requiring precomputations of $3P, 5P, \ldots, (2^{w-1}-1)P$ and $3Q, 5Q, \ldots, (2^{w-1}-1)Q$. More recently, Doche et al. introduced joint double-base chains to represent a pair of numbers in [24]. The advantage of having a joint representation for a pair of scalars is that we can apply Straus' idea [25] (also known as Shamir's trick) to combine point doublings.

The different number representations that we discussed above consider techniques which minimize the number of nonzero elements or columns in the representation. However, we can also reduce the length of these representations, for example, by using higher radices. The hybrid binary-ternary number system (HBTNS) [26] can be used to find a

- J. Adikari and V.S. Dimitrov are with the Advanced Technology Information Processing Systems (ATIPS), Department of Electrical and Computer Engineering, The University of Calgary, 2500 University Dr. NW, Calgary, AB T2N 1N4, Canada. E-mail: {jithra.adikari, dimitrov}@atips.ca.
- L. Imbert is with the CNRS-PIMS, CISaC, Department of Mathematics and Statistics, The University of Calgary, Canada, and the CNRS-LIRMM, Université Montpellier 2, 161 rue Ada, F-34095 Montpellier, France. E-mail: laurent.imbert@lirmm.fr.

Manuscript received 22 Aug. 2009; revised 17 Mar. 2009; accepted 19 Apr. 2010; published online 10 June 2010.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2009-08-0405.
Digital Object Identifier no. 10.1109/TC.2010.138.

1. During the final preparation of this manuscript, Patrick Longa pointed out that w-HBTF is very similar to the Extended-wmbNAF method that he proposed in his MS thesis.

short and sparse representation for a single scalar or a joint representation for a pair of scalars. Note that the HBTNS is a special case of double-base representation. More precisely, any given integer is having one or more double-base representations. Double-base chains are special cases of double-base representations, while HBTNS corresponds to a particular double-base chain for a given integer.

In this paper, which is a substantial extension of the work presented at the ARITH 19 Symposium [27], we propose three novel algorithms based on the HBTNS, namely, the window hybrid binary-ternary form (w-HBTF) for single-point multiplication, and the hybrid binary-ternary joint form (HBTJF) and reduced hybrid binary-ternary joint form (RHBTJF) for double-point multiplication.

In the case of generic short Weierstrass curves, we achieve up to 12 percent improvement in single-scalar multiplication and up to eight percent gain in double-scalar multiplication. However, over tripling-oriented DIK curves, a family of curves which allows for fast point tripling, our software implementation is faster than w-NAF by more than 25 percent. One may argue that these curves are "special," and, therefore, provide less security. The belief that random curves provide more security than specific curves has been seriously criticized by Koblitz et al. in [28]. Their conclusion is that, in some cases, random curves may not provide the level of security one would think. Therefore, using well chosen specific curves, such as DIK-3 curves, does not weaken a cryptosystem.

The organization of this paper is as follows: In Section 2, the basics of elliptic curve and hybrid binary-ternary number system are presented. In Section 3, we extend the hybrid binary-ternary concept to represent a scalar with a novel recoding algorithm. In Section 4, we describe two novel algorithms for double-scalar multiplication. We present our software implementation and some numerical comparisons in Section 5 and conclude the paper in Section 6.

2 Background

2.1 Elliptic Curve Arithmetic

The general definition of an elliptic curve E defined over a field $\mathbf{K}$ is given by the Weierstrass equation

$$E/\mathbf{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbf{K}$. In addition, the elliptic curve must be smooth, i.e., there is no point on the curve which has two or more distinct tangent lines. Smooth curves can be constructed by carefully selecting the coefficients a_1, a_2, a_3, a_4, a_6 in (1). (See [5] for more details).

For fields of characteristic not equal to 2 or 3, we use the short Weierstrass equation

$$E/\mathbf{K} : y^2 = x^3 + ax + b. \tag{2}$$

The coefficients a, b, and the underlying field $\mathbf{K}$ can be selected to optimize the efficiency of the elliptic curve operations (e.g., choosing $a = -3$ allows for faster point arithmetic).

In [29], Doche et al. introduced a new family of elliptic curves that are very efficient for point triplings. Tripling-oriented Doche-Ichart-Kohel (DIK) curves are defined over a field of characteristic larger than three and have a rational three-torsion subgroup. They can be expressed as

$$E/\mathbf{K} : y^2 = x^3 + 3u(x+1)^2. \tag{3}$$

Over DIK-3 curves, the cost ratio (in terms of field operations) between a point tripling and a point doubling is smaller than the same ratio over other known models or families of elliptic curves.

When point addition, doubling, and tripling operations are executed in affine coordinates, field inversions are required. Field inversion is significantly expensive compared to other field arithmetic operations, namely, addition, subtraction, and multiplication. To minimize field inversions in our computations, we use projective coordinates.

Let $\mathbf{K}$ be the field over which the elliptic curve is defined. The set of affine coordinates, $\mathbf{A}(\mathbf{K})$ is given by

$$\begin{aligned}\mathbf{A}(\mathbf{K}) = \{(x,y) \in \mathbf{K} \times \mathbf{K} : y^2 + a_1xy + a_3y \\ - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \infty,\end{aligned} \tag{4}$$

where ∞ is the point at infinity. Let c and d be positive integers and let $x = X/Z^c$ and $y = Y/Z^d$, then we define the projective coordinates as

$$\mathbf{P}(\mathbf{K}) = \{(X : Y : Z) : X, Y, Z \in \mathbf{K}, Z \neq 0\}. \tag{5}$$

The set of projective points at infinity is defined by

$$\mathbf{P}(\mathbf{K})^0 = \{(X : Y : Z) : X, Y, Z \in \mathbf{K}, Z = 0\}. \tag{6}$$

The set $\mathbf{P}(\mathbf{K})^0$ is called the line at infinity because its points do not correspond to any of the affine points. We define standard projective coordinates by setting $c = 1$ and $d = 1$. In addition to projective coordinates, Jacobian coordinates are defined when $c = 2$ and $d = 3$ [5], [12], [30], [31].

2.2 Hybrid Binary-Ternary Number System

Dimitrov and Cooklev introduced the hybrid binary-ternary number system in 1995 in [26] in order to speed up modular exponentiation. The proposed method for computing this particular recoding of an integer is illustrated in Algorithm 1.

Algorithm 1. HBTNS representation
Input: An integer $n > 0$
Output: Arrays `digits[], base[]`

```
i = 0
while n > 0 do
    if n ≡ 0 (mod 3) then
        digits[i] = 0
        base[i] = 3
    else if n ≡ 0 (mod 2) then
        digits[i] = 0
        base[i] = 2
    else
        digits[i] = 1
        base[i] = 2
    end if
    n = ⌊n/ base[i] ⌋
    i = i + 1
end while
return digits[], base[]
```

Mixing bases two and three in the representation of n can be seen as expressing n in a base that is a real number between 2 and 3. Using some probabilistic arguments, this

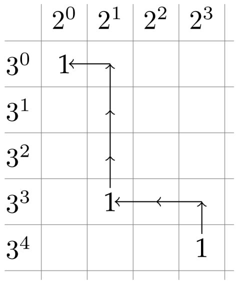

Fig. 1. An example of staircase walk for a double-base chain representing 703.

average base β can be easily evaluated. For the recoding algorithm presented in Algorithm 1, one obtains $\beta = 2^{10/13}3^{3/13} \approx 2.1962$. Consequently, a t-bit integer has $(log_\beta 2)t \approx 0.8811t$ digits on average. This corresponds to a reduction of roughly 12 percent compared to the binary length. The proportion of ones can also be evaluated to $5/13 \approx 0.3846$, that is approximately $0.3388t$.

Example 1. The hybrid binary-ternary form of $n = 703 = (1010111111)_2$ given by

$$\texttt{digits}[\,] = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 1]$$
$$\texttt{base}[\,] = [2\ 2\ 3\ 2\ 3\ 3\ 3\ 2]$$

has only eight digits among which three only are nonzero. In standard binary representation, 703 is 10-bit long and has eight nonzero bits. Note that the least significant digit is the right-most value in `digits[]`, such that $703 = 2^3 3^4 + 2^1 3^3 + 1$.

Using a 2D array, we can visualize the expansion of the hybrid binary-ternary representation of an integer. In this graphic representation, we use powers of two in columns and powers of three in rows. Powers of two grow from left to right while powers of three grow downward making the upper-left corner $2^0 3^0 = 1$. A staircase walk from the bottom-right nonzero element to the top-left nonzero element gives the corresponding double-base chain generated by Algorithm 1. The convention in the staircase representation is going up as much as possible before going left. We give an example of staircase walk in Fig. 1.

A similar idea has been introduced in the context of ECC by Ciet et al. in [32]. Dimitrov et al. generalized this concept in [13] by using a greedy approach to compute special signed double-base expansions [33], [34], i.e., expressions of the form

$$\sum_i \pm 2^{a_i} 3^{b_i}, \quad \text{with } a_i, b_i \geq 0,$$

where the exponents form two simultaneously decreasing sequences. These expansions, called double-base chains (see Definition 1 below), allow for fast scalar multiplication. See [14], [35], [36], [37] for more details about this number system and double-base chain generation.

Definition 1 (Double-base chain). *Given $k > 0$, a sequence $(K_n)_{n>0}$, of positive integers satisfying: $K_1 = 1$, $K_{n+1} = 2^u 3^v K_n + s$, with $s \in \{-1, 1\}$ for some $u, v \geq 0$, and such that $K_m = k$ for some $m > 0$, is called a double-base chain for k. The length m of a double-base chain is equal to the number of terms (often called $\{2,3\}$-integers), used to represent k.*

Later, Doche and Imbert introduced window-based double-base chains mixing both concepts of double-base chains and w-NAF in [15]. The scalar multiplication can be speeded up by using fast addition, doubling and tripling formulas in different coordinates.

3 Single Scalar Multiplication

3.1 Window Hybrid Binary-Ternary Form

In this section, we introduce the window hybrid binary-ternary form (w-HBTF) for single scalar multiplication. Extending the concept of w-NAF where w represents the width of a 1D window, the value of w in w-HBTF is an expression of the form $2^b 3^t$ with $b, t \in \mathbf{N}$, which can be seen as a 2D window of width b and height t. For example, when $b = 1$ and $t = 2$ we get a window of size $2^1 3^2 = 18$. Note that when $t = 0$ the hybrid binary-ternary representation is equivalent to 2^b-NAF.

Algorithm 2 is an extension of Algorithm 1. We start by checking whether the input number is divisible by 2 or 3 and, if this is the case, we assign the corresponding digit to zero and the base accordingly. If the number k is neither divisible by 2 nor 3, we subtract $k \bmod\text{s}\ w$ from k such that the result is divisible by $w = 2^b 3^t$. The corresponding digit is set to $k \bmod\text{s}\ w$, an integer in $[-2^{b-1}3^t, 2^{b-1}3^t]$, while base is set to 3. The value k is then divided by 3. This guarantees that the next $t + b - 1$ digits will all be zero.

Algorithm 2. w-hybrid binary-ternary form (w-HBTF)
Input: A positive integer k, two integers $b, t > 0$ such that $w = 2^b 3^t$
Output: Arrays `whbt[]`, `base[]`

```
i = 0
while k > 0 do
    if k ≡ 0 (mod 2) then
        whbt[i] = 0
        base[i] = 2
    else if k ≡ 0 (mod 3) then
        whbt[i] = 0
        base[i] = 3
    else
        whbt[i] = k mods 2^b 3^t
        base[i] = 2
        k = k − whbt[i]
    end if
    k = k/base[i]
    i = i + 1
end while
return whbt[], base[]
```

Example 2. In the following example, we give the 4-NAF and 5-NAF decompositions of 727:

$$4\text{-NAF}(727) = [0\ 0\ 3\ 0\ 0\ 0\ \bar{3}\ 0\ 0\ 0\ 7],$$
$$5\text{-NAF}(727) = [1\ 0\ 0\ 0\ 0\ \bar{9}\ 0\ 0\ 0\ 0\ \bar{9}],$$

and its 12 and 18-HBTF

TABLE 1
Theoretical Comparison of w-NAF and Various w-HBTF for a t-bit Integer

Avg. values	w-NAF	6-HBTF	12-HBTF	18-HBTF	24-HBTF	36-HBTF
base	2	2.38	2.29	2.51	2.23	2.40
length	$t+1$	$0.80t$	$0.84t$	$0.75t$	$0.86t$	$0.79t$
# base 2 digits	$t+1$	$0.46t$	$0.56t$	$0.63t$	$0.34t$	$0.43t$
# base 3 digits	0	$0.34t$	$0.28t$	$0.42t$	$0.24t$	$0.36t$
# non-zero digits	$t/(w+1)$	$0.23t$	$0.19t$	$0.17t$	$0.16t$	$0.14t$
Precomp.	$2^{w-2}-1$	0	1	2	3	5

$$\begin{aligned}\text{12-HBTF}(727) &= [5\ 0\ 0\ 1\ 0\ 0\ \bar{5}]\\ \texttt{base}[\,] &= [2\ 3\ 2\ 2\ 3\ 2\ 2],\\ \text{18-HBTF}(727) &= [5\ 0\ 0\ 0\ 0\ 0\ 7]\\ \texttt{base}[\,] &= [2\ 3\ 3\ 2\ 2\ 2\ 2].\end{aligned}$$

Just like w-NAF, the use of w-HBTF for elliptic-curve-scalar multiplication requires some precomputed points. For $w = 2^b 3^t$, these points are of the form $[d]P$ with $-2^{b-1}3^t \leq d \leq 2^{b-1}3^t$ and $\gcd(d, 2, 3) = 1$. For instance, for 18-HBTF only the points $\pm 5P$ and $\pm 7P$ are required. Because the negation of a point is easy to compute, only one of $\pm 5P$ and one of $\pm 7P$ are precomputed and stored in this case.

3.2 Theoretical Analysis of w-HBTF

Algorithm 2 is designed to produce a more compact (fewer digits) and sparser (fewer nonzero digits) representation than the binary, NAF, and w-NAF recoding schemes. These parameters can be precisely estimated using probabilistic arguments. Considering a Markov process, we can deduce the probability to divide a number by 2 or by 3 and the probability to get a zero or a nonzero digit.

We build a transition graph with w states, where each state corresponds to a residue class modulo w. The corresponding $w \times w$ transition matrix contains the probabilities to go from state i to state i', i.e., the probabilities to obtain a number of the form $wt' + i'$ from a number of the form $wt + i$ after performing either a division by 2 or by 3 or a subtraction of i followed by a division by 2. More precisely, if the current number, say, $wt + i$ is divisible by 3, we obtain with probability $1/2$, either a number of the form $wt' + i/2$ or a number of the form $wt' + w/2 + i/2$ depending on the parity of t. Similarly, if the current number is divisible by 3, we get with probability $1/3$ a number of the form $wt' + i/3$ or $wt' + w/3 + i/3$ or $wt' + 2w/3 + i/3$. If none of the above conditions are satisfied, we make the current number divisible by w by subtracting the suitable value and we divide it by 3. Therefore, we obtain with probability $1/2$, a number of the form wt' or $wt' + w/2$.

If $P_{i,i'}$ denotes the probability to go from state i to state i', we obtain the following probabilities:

- if $i \equiv 0 \pmod 2$ then
  $P_{i,i'} = \frac{1}{2}$ for $i' \in \{\frac{i}{2}, \frac{i}{2} + \frac{w}{2}\}$ and 0 otherwise.
- if $i \equiv 0 \pmod 3$ then
  $P_{i,i'} = \frac{1}{3}$ for $i' \in \{\frac{i}{3}, \frac{i}{3} + \frac{w}{3}, \frac{i}{3} + \frac{2w}{3}\}$ and 0 otherwise.
- if $\gcd(i, 2, 3) = 1$ then
  $P_{i,i'} = \frac{1}{2}$ for $i' \in \{0, \frac{w}{2}\}$ and 0 otherwise.

For example, for $w = 6$ the above relations lead to the following transition matrix:

$$M_S = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0\\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0\\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0\\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3}\\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2}\\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix}.$$

The stationary distribution is then calculated as

$$\pi_\infty = \lim_{n\to\infty} \pi_0 M_S^n,$$

with $\pi_0 = (1/6,\ 1/6,\ 1/6,\ 1/6,\ 1/6,\ 1/6)$ the initial probabilities (although they do not play any role). We obtain

$$\pi_\infty = (2/7,\ 1/7,\ 0,\ 3/7,\ 0,\ 1/7).$$

We deduce the average probabilities: p_2 (respectively p_3) to perform a division by 3 (respectively 3) and p_{nz} (respectively p_z) to get a nonzero (respectively zero) digit:

$$p_3 = \pi_\infty[3] = \frac{3}{7}, \qquad p_2 = 1 - p_3 = \frac{4}{7},$$
$$p_{nz} = \pi_\infty[1] + \pi_\infty[5] = \frac{2}{7}, \quad p_z = 1 - p_{nz} = \frac{5}{7}.$$

The average base β for 6-HBTF can be evaluated by

$$\beta = 2^{\frac{4}{7}}3^{\frac{3}{7}} = \sqrt[7]{2^4 3^3} = \sqrt[7]{432} \approx 2.379565578968.$$

Therefore the average length of 6-HBTF is given by

$$(\log_\beta 2) \times t \approx 0.86691794 \times t$$

for a t-bit number. Finally, we evaluate its density of nonzero digits with

$$0.86691794 \times t \times \frac{2}{7} \approx 0.24769084028 \times t.$$

Using a similar analysis, we have computed these quantities for different window sizes. The results are summarized in Table 1. In the last row, we give the number of precomputations required for single scalar multiplication.

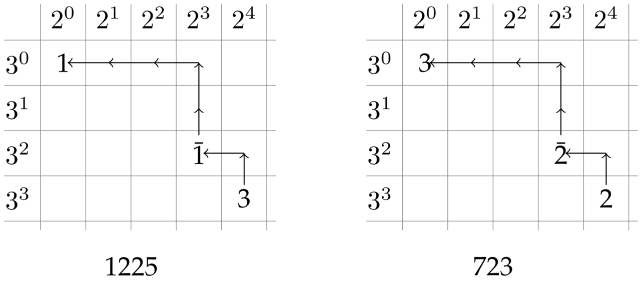

Fig. 2. Double-base chains for 1,225 and 723 obtained using Algorithm 3.

4 Double Scalar Multiplication

In this section, we present two algorithms for double scalar multiplication based on HBTNS, namely, the Hybrid Binary-Ternary Joint Form (HBTJF) and Reduced Hybrid Binary-Ternary Joint Form (RHBTJF).

4.1 Hybrid Binary-Ternary Joint Form

4.1.1 Algorithm

In this new joint number representation, both scalars share the same base sequence which, as before, mixes integers 2 and 3. In the following, we denote by column a triple (b_i, d_i, d_i'), where b_i is the common base used for the ith pair of digits d_i and d_i'. If both d_i and d_i' are zero, we call it a zero column; otherwise, it is a nonzero column. Our algorithm is designed to generate fewer nonzero columns than its counterparts. We describe our new joint representation in Algorithm 3.

Algorithm 3. Hybrid binary-ternary joint form
Input: Two positive integers k_1, k_2
Output: Arrays `hbt1[], hbt2[], base[]`

$i = 0$
while $k_1 > 0$ or $k_2 > 0$ **do**
  if $k_1 \equiv 0 \pmod 2$ and $k_2 \equiv 0 \pmod 2$ **then**
    $\texttt{hbt1}[i] = 0$, $\texttt{hbt2}[i] = 0$
    $\texttt{base}[i] = 2$
  else if $k_1 \equiv 0 \pmod 3$ and $k_2 \equiv 0 \pmod 3$ **then**
    $\texttt{hbt1}[i] = 0$, $\texttt{hbt2}[i] = 0$
    $\texttt{base}[i] = 3$
  else
    $\texttt{hbt1}[i] = k_1 \text{ mods } 6$, $\texttt{hbt2}[i] = k_2 \text{ mods } 6$
    $\texttt{base}[i] = 2$
    $k_1 = k_1 - \texttt{hbt1[i]}$, $k_2 = k_2 - \texttt{hbt2[i]}$
  end if
  $k_1 = k_1/\texttt{base}[i]$, $k_2 = k_2/\texttt{base}[i]$
  $i = i + 1$
end while
return `hbt1[], hbt2[], base[]`

The computation of the HBTJF for two scalars k_1, k_2 starts by checking whether both numbers k_1 and k_2 are divisible by 2. If it is the case, the common base is set to two and both digits are set to 0. In other words, a zero column in base 2 is generated. Failing this first condition, both k_1 and k_2 are checked for divisibility by three. If both numbers are divisible by three, then a zero-column in base 3 is produced. If none of the above conditions are satisfied, i.e., if k_1 and k_2 are neither divisible by 2 nor 3 simultaneously, then the values k_i mods 6 for $i = 1, 2$ are subtracted from k_1 and k_2, respectively, such that both scalars become simultaneously divisible by 6. We then divide both numbers by 2. This step generates a nonzero column in base 2 with the guarantee to generate a zero-column in base 3 at the next step. We repeat this procedure until both k_1 and k_2 are equal to 0. Note that in the case of a nonzero column, the possible digits belong to the set $\{-2, -1, 0, 1, 2, 3\}$.

TABLE 2
HBTJF Precomputations

	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. The following example illustrates the potential advantage of the HBTJF over interleaving method. For $k_1 = 1{,}225$ and $k_2 = 723$, the interleaving method with different window sizes five and four leads to the following decompositions:

$$\text{5-NAF}(1,225) = [1\ 0\ 0\ 0\ 0\ \overline{13}\ 0\ 0\ 0\ 0\ 0\ 9],$$
$$\text{4-NAF}(723) = [0\ 0\ 0\ 3\ 0\ 0\ 0\ \bar{3}\ 0\ 0\ 0\ 3],$$

which have six nonzero elements. Note that when the interleaving method is used for double-scalar multiplication, the cost depends on the number of nonzero elements instead of the number of nonzero columns since it would be too expensive to precompute all the possible combinations of points which could occur for a column. Larger window sizes are, therefore, possible. (For this example, the decomposition using $w = 4$ for 1,225 and $w = 5$ for 723 also leads to six nonzero digits.) Using Algorithm 3, the hybrid binary-ternary joint form given below

$$1,225 = [3\ 0\ \bar{1}\ 0\ 0\ 0\ 0\ 1],$$
$$723 = [2\ 0\ \bar{2}\ 0\ 0\ 0\ 0\ 3],$$
$$\texttt{base}[\,] = [2\ 3\ 2\ 2\ 2\ 3\ 3\ 2],$$

only requires eight digits (as opposed to 12 above) and has only three nonzero columns. The corresponding double-base chains are shown in Fig. 2. Note that both chains share the same staircase walk, a consequence of expressing both numbers with a single base sequence. In this example, however, the digits also occur exactly at the same location. Note that it is not necessarily the case and only occurs here because every nonzero columns have both digits different from zero.

The digits obtained using Algorithm 3 belong to the set $\{-2, -1, 0, 1, 2, 3\}$. This leads to 14 points that need to be precomputed online. These points are given in Table 2. 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. Since the negation of a point is negligible, only one set of point difference needs to be calculated; for example, $2Q - 3P$ is easily obtained from $3P - 2Q$. In contrast, interleaving 5-NAF needs seven offline precomputations for the point that is known in advance, plus another seven online precomputations for the other point.

4.1.2 Theoretical Analysis of HBTJF

In this section, we analyze Algorithm 3. Our aim is to evaluate the probabilities of occurrence of base 2 and base 3 in the hybrid joint representation. As in the single-scalar case, we calculate the average base and deduce the proportion of nonzero columns.

We consider classes of congruence modulo 6 for a pair of integers. We have 36 different state, denoted S_{i_1,i_2}, corresponding to the 36 distinct pairs of integers of the form $(6k_1 + i_1, 6k_2 + i_2)$ for $k_1, k_2 \in \mathbf{N}$ and $i_1, i_2 \in \{0, 1, 2, 3, 4, 5\}$. We construct the 36×36 transition matrix M_D, where the entries correspond to the probabilities to go from state S_{i_1,i_2} to state S_{j_1,j_2} after applying one step of Algorithm 3.

If both numbers are even, divisions by 2 (performed in step 14) lead to any of $S_{0,0}$, $S_{0,3}$, $S_{3,0}$, $S_{3,3}$ with probability $1/4$. Similarly, the state $S_{3,0}$ corresponds to the case where both numbers are divisible by 3 but not by 2. The divisions by 3 lead to any of the states $S_{1,0}$, $S_{1,2}$, $S_{1,4}$, $S_{3,0}$, $S_{3,2}$, $S_{3,4}$, $S_{5,0}$, $S_{5,2}$, $S_{5,4}$, with probability $1/9$. In the last case, i.e., when both numbers are neither divisible by 3 nor 2 simultaneously, we subtract the suitable values to obtain a pair of number simultaneously divisible by six and we perform a division by 2. Hence, we reach one of the four states $S_{0,0}$, $S_{0,3}$, $S_{3,0}$, $S_{3,3}$ with probability $1/4$. These states correspond to the four pairs of multiples of 3. The complete transition matrix M_D is given in the Appendix.

As before, the stationary distribution π_∞ is equal to $\lim_{n\to\infty} \pi_0 M_D^n$, where $\pi_0 = (1/36, \ldots, 1/36)$ denotes the initial probabilities (although they do not play any role). We have

$$\pi_\infty[i] = \begin{cases} \dfrac{8}{59}, & \text{for } i = 0, \\ \dfrac{9}{59}, & \text{for } i \in \{3, 18, 21\}, \\ 0, & \text{for } i \in \{2, 4, 12, 14, 16, 24, 26, 28\}, \\ \dfrac{1}{59}, & \text{otherwise.} \end{cases}$$

This allows us to compute the following average probabilities:

$$p_3 = \sum_{i,j \in \mathcal{S}_3} \pi_\infty[6i + j],$$

where $\mathcal{S}_3 = \{i, j \equiv 0 \pmod 3 \text{ and i,j} \not\equiv 0 \pmod 2\}$, and

$$p_z = \sum_{i,j \in \mathcal{S}_z} \pi_\infty[6i + j],$$

where $\mathcal{S}_z = \{i, j \equiv 0 \pmod 3 \text{ or i,j} \equiv 0 \pmod 2\}$. As before, p_3 denotes the probability to perform a division by 3 and p_z denote the probability to generate a zero column. Clearly, the probability to perform a division by 2 is $p_2 = 1 - p_3$ and the probability to generate a nonzero column is $p_{nz} = 1 - p_z$. We have

$$p_2 = \frac{32}{59}, \quad p_3 = \frac{27}{59}, \quad p_z = \frac{35}{59}, \quad p_{nz} = \frac{24}{59}. \qquad (7)$$

TABLE 3
Theoretical Comparison of HBTJF and Interleaving w-NAF for a Pair of t-bit Integers

Avg. values	HBTJF	Int. 4-NAF	Int. 5-NAF
base	2.41	2	2
# col.	$0.79t$	$t+1$	$t+1$
# base 2	$0.43t$	$t+1$	$t+1$
# base 3	$0.36t$	0	0
# non-zero	$0.32t$	$0.4t$	$0.33t$
Precomp.	14	6	14

Now, using p_2 and p_3, we can evaluate the average base

$$\beta = \sqrt[59]{2^{32}3^{27}} = \sqrt[59]{32751691810479015985152} \approx 2.407765.$$

For a pair of t-bit integers, the average number of columns in the HBTJF is approximately

$$(\log_\beta 2) \times t \approx 0.7888 \times t. \qquad (8)$$

Finally, from (7) and (8), we derive that the expected number of elliptic curve additions per bit is approximately

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

We summarize our theoretical results and compare them to interleaving w-NAF in Table 3, with real values rounded to the nearest hundredth. For simplicity, we consider that the same window width is used for both numbers in the interleaving w-NAF method.

4.2 Reduced Hybrid Binary-Ternary Joint Form

4.2.1 Algorithm

As shown in Table 2, the hybrid binary-ternary joint form needs 14 online precomputations, which may not be acceptable for devices with limited memory. In this section, we propose the reduced HBTJF, which reduces the number of online precomputations to two points, namely, $P + Q$ and $P - Q$. The decomposition method is presented in Algorithm 4.

Algorithm 4. Reduced hybrid binary-ternary joint form
Input: Two positive integers k_1, k_2
Output: Arrays `rhbt1[], rhbt2[], base[]`

$i = 0$
while $k_1 > 0$ or $k_2 > 0$ **do**
  if $k_1 \equiv 0 \pmod 2$ and $k_2 \equiv 0 \pmod 2$ **then**
    $\texttt{rhbt1}[i] = 0$, $\texttt{rhbt2}[i] = 0$
    $\texttt{base}[i] = 2$
  else if $k_1 \equiv 0 \pmod 3$ and $k_2 \equiv 0 \pmod 3$ **then**
    $\texttt{rhbt1}[i] = 0$, $\texttt{rhbt2}[i] = 0$
    $\texttt{base}[i] = 3$
  else
    if $k_1 \equiv 0 \pmod 4$ or $k_2 \equiv 0 \pmod 4$ **then**
      $\texttt{rhbt1}[i] = k_1 \text{ mods } 4$,
      $\texttt{rhbt2}[i] = k_2 \text{ mods } 4$
      $\texttt{base}[i] = 2$

13: **else**
14: $\texttt{rhbt1}[i] = k_1 \text{ mods } 3$,
$\texttt{rhbt2}[i] = k_2 \text{ mods } 3$
15: $\texttt{base}[i] = 3$
16: **end if**
17: $k_1 = k_1 - \texttt{rhbt1}[i]$, $k_2 = k_2 - \texttt{rhbt2}[i]$
18: **end if**
19: $k_1 = k_1/\texttt{base}[i]$, $k_2 = k_2/\texttt{base}[i]$
20: $i = i + 1$
21: **end while**
22: **return** `rhbt1[], rhbt2[], base[]`

The difference with Algorithm 3 which computes the (nonreduced) HBTJF is in the treatment of the last condition, that is, when k_1 and k_2 are neither divisible by 2 or by 3 simultaneously (steps 9-18). Instead of subtracting a value from $\{-2, \ldots, 3\}$ from both numbers to get a pair of integers that is divisible by 6, we now check whether k_1 or k_2 is divisible by 4. If so, we subtract $k_1 \text{ mods } 4$ and $k_2 \text{ mods } 4$ from k_1 and k_2, respectively, followed by a division by 2. Finally, if none of the above conditions are satisfied, we subtract $k_1 \text{ mods } 3$ and $k_2 \text{ mods } 3$ from k_1 and k_2 and perform a division by 3. We reiterate the whole procedure until both k_1 and k_2 are zero.

Example 4. In the following example, we compare Solinas' JSF with the RHBTJF since the precomputations are identical in both methods. The JSF of 1,225 and 723 has 11 columns out of which seven are nonzero

$$\begin{aligned} 1{,}225 &= [1\ 0\ 1\ 0\ \bar{1}\ 0\ 0\ 1\ 0\ 0\ 1], \\ 723 &= [1\ 0\ \bar{1}\ 0\ 0\ \bar{1}\ 0\ \bar{1}\ \bar{1}\ 0\ \bar{1}]. \end{aligned}$$

The reduced hybrid binary-ternary joint form obtained from Algorithm 4 has length nine with five nonzero columns

$$\begin{aligned} 1{,}225 &= [1\ \bar{1}\ 0\ \bar{1}\ 0\ 0\ 0\ 0\ 1], \\ 723 &= [0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0], \\ \texttt{base}[\,] &= [2\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 3]. \end{aligned}$$

4.2.2 Theoretical Analysis of RHBTJF

First of all, let us prove that the only precomputations are indeed $P + Q$ and $P - Q$. Clearly, the first two conditions generate zero columns and no point is added in the course of a scalar multiplication. The first nonzero column may occur if either k_1 or k_2 is congruent to 0 modulo 4. In this case, the digits are set to $k_i \text{ mods } 4$ for $i = 1, 2$, i.e., a value in $\{-1, 0, 1, 2\}$ in theory. However, if the condition is satisfied because one of the two numbers is divisible by 4, we know that the other number is not divisible by 2. Since in that case, both numbers are divisible by 2 and the first condition (in step 3) would have been satisfied. Therefore, digit 2 never occurs. If none of the first three conditions are satisfied, the digits are obtained by computing $k_i \text{ mods } 3$ for $i = 1, 2$ and clearly belong to $\{-1, 0, 1\}$.

As in Section 4.1.2, we can evaluate different relevant probabilities using a simple Markov process. In this case, we need to consider residue classes modulo 12 for a pair of integer, which lead to a 144×144 transition matrix M_{RD} (not given in this paper).

TABLE 4
Theoretical Comparison of RHBTJF, JSF, and JDBC for a Pair of t-bit Integers

Parameters	RHBTJF	JSF	JDBC
Average base	2.37	2	–
Avg # col.	$0.80t$	$t+1$	–
Avg # base 2 col.	$0.47t$	$t+1$	–
Avg # base 3 col.	$0.33t$	0	–
Avg # non-zero col.	$0.43t$	$0.5t$	–
Precomp.	2	2	2

Starting from initial probabilities $\pi_0 = (1/144, \ldots, 1/144)$, the stationary distribution is obtained by computing $\pi_\infty = \lim_{n\to\infty} \pi_0 M_{RD}^n$ as before. This gives us p_z, p_{nz} the probabilities of a zero and nonzero column, respectively, and p_2, p_3 the probabilities of performing a division by 2 and 3, respectively. We have:

$$\begin{aligned} p_z &= \frac{272}{583} \approx 0.4665523156, \\ p_{nz} &= \frac{311}{583} \approx 0.5334476844, \\ p_2 &= \frac{832}{1421} \approx 0.5855031668, \\ p_3 &= \frac{589}{1421} \approx 0.4144968332. \end{aligned} \tag{9}$$

We evaluate the average base

$$\beta = 2^{832/1421} 3^{589/1421} \approx 2.366024518.$$

Therefore, the average length of the representation is given by:

$$(\log_\beta 2) \times t \approx 0.8048516306 \times t. \tag{10}$$

From (9) and (10), we get the average number of nonzero columns in the RHBTJF for a pair of t-bit integers

$$\frac{311}{583} \times 0.8048516306 \times t \approx 0.4293462386 \times t.$$

In Table 4, we summarize these results and compare them with Solina's jsf and Doche's joint double-base chains [37] since all three methods require the same precomputations.

5 Implementation and Results

In this section, we present experimental results based on our software implementation and we compare our algorithms to their counterparts. We carried out a software implementation on two kinds of curves:

- Short Weierstrass curves with $a = -3$,
- Tripling-oriented Doche-Ichart-Kohel (DIK3) curves.

Choosing $a = -3$ in the equation of a short Weierstrass curves allows some savings in repeated point doubling [38]. We considered the curves recommended by the NIST [39] defined over finite fields of prime characteristic of sizes 192, 224, 256, 384, and 521 bits, respectively.

TABLE 5
Comparison of 6-HBTF, NAF, and DB-Chains for DIK3 Curves

Size	6-HBTF	NAF		DB-Chains	
192	2.41	3.07	(+27%)	3.45	(+43%)
224	3.17	3.95	(+25%)	4.52	(+43%)
256	3.85	4.79	(+24%)	5.75	(+49%)
384	8.88	11.00	(+24%)	13.52	(+52%)
521	17.48	21.30	(+22%)	20.19	(+16%)

TABLE 6
Comparison of 6-HBTF, NAF, and DB-chains for Weierstrass curves

Size	6-HBTF	NAF		DB-Chains	
192	2.53	2.83	(+12%)	3.45	(+36%)
224	3.31	3.66	(+11%)	4.61	(+39%)
256	4.12	4.54	(+10%)	5.80	(+41%)
384	9.48	10.47	(+10%)	14.02	(+48%)
521	17.86	19.48	(+9%)	25.82	(+45%)

TABLE 7
Comparison of 18-HBTF, 12-HBTF, and 3-NAF for DIK3 Curves

Size	18-HBTF	12-HBTF		3-NAF	
192	2.27	2.43	(+7%)	2.87	(+26%)
224	2.96	3.16	(+7%)	3.73	(+26%)
256	3.59	3.85	(+7%)	4.52	(+26%)
384	8.20	8.78	(+7%)	10.21	(+25%)
521	16.19	17.27	(+7%)	19.90	(+23%)

TABLE 8
Comparison of 18-HBTF, 12-HBTF, and 3-NAF for Weierstrass Curves

Size	18-HBTF	12-HBTF		3-NAF	
192	2.47	2.49	(+1%)	2.62	(+6%)
224	3.22	3.25	(+1%)	3.42	(+6%)
256	3.99	4.02	(+1%)	4.24	(+6%)
384	9.11	9.18	(+1%)	9.67	(+6%)
521	17.08	17.32	(+1%)	18.17	(+6%)

TABLE 9
Comparison of 18-HBTF, 24-HBTF, and 4-NAF for DIK3 Curves

Size	18-HBTF	24-HBTF		4-NAF	
192	2.27	2.47	(+9%)	2.81	(+24%)
224	2.96	3.20	(+8%)	3.56	(+20%)
256	3.59	3.86	(+8%)	4.37	(+22%)
384	8.20	8.71	(+6%)	9.83	(+20%)
521	16.19	17.09	(+6%)	19.11	(+18%)

The same fields are used for DIK3 curves. These curves allow for very fast point tripling [29]. According to [40], a small value or some power of two is a good choice for u in (3). For our benchmarks, we considered the simpler case $u = 2$, i.e., our DIK3 curves are defined by the equation $y^2 = x^3 + 6(x + 1)^2$.

Regarding coordinates, we used Jacobian coordinates for short Weierstrass curves ($x = X/Z^2,\ y = Y/Z^3$) and modified Jacobian coordinates ($x = X/Z,\ y = Y/ZZ,\ ZZ = Z^2$) for DIK3 curves.

For single scalar multiplication, the program first calculates a random scalar k and a random point P, on the curve. These calculations are not taken into account in our timings. The necessary precomputations are performed online and the time required to compute them is naturally taken into account for our comparisons.

For double scalar multiplication, the program initially generates a pair of random scalars (k_1, k_2) and two points on the curve. We consider that one point, say P is known in advance. Therefore, the precomputations involving only P are performed offline and are not taken into account in our timings. All the other precomputations are performed online.

For both single and double scalar multiplication, the precomputed points are converted in affine coordinate in order to save some field operations when additions involving those points occur. (The operation which consist in adding a point in Jacobian coordinate to a point in affine coordinate, i.e., with $Z = 1$, is called a mixed addition). To compute the affine coordinates, we need to compute the inverse of the Z coordinates. Since inversion is a costly operation, we use Montgomery trick [41], [42]: we multiply all Z coordinates together and perform only one inversion for the product. For example, with two integers Z_1 and Z_2, one can compute $Z_1^{-1} = Z_2/Z_1Z_2$ and $Z_2^{-1} = Z_1/Z_1Z_2$ with one inversion and three multiplications, instead of two inversions.

Finally, in the case of HBTJF, the precomputations of pairs of points of the form $(aP + bQ, aP - bQ)$ can be optimized by reusing some partial results. For example, it is possible to compute both $P + Q$ and $P - Q$ with five multiplications and three squarings, instead of eight multiplications and four squarings if the two points are computed separately.

Our software implementation was implemented in `C++` with the GNU Multiple Precision (GMP) library version `4.2.2` [43]. The binaries have been compiled with `g++` version `4.1.3`. Our benchmarks ran on an AMD Sempron 1.8 GHz with 1 GB memory. Timings are given in millisecond per scalar multiplication (either single or double).

In Tables 5, 6, 7, 8, 9, 10, 11, 12, we compare several single-scalar multiplication algorithm. We present fair comparisons based on the amount of precomputations required by each algorithm. For example, we compare 4-NAF with 24-HBTF because both representations need three precomputed points ($3P$, $5P$, and $7P$ for 4-NAF and $5P$, $7P$, and $11P$ for 24-HBTF). When the amount of precomputations required by w-HBTF does not match any window-NAF, we propose two close comparisons. For example, 18-HBTF, which needs two precomputations is compared to both 3-NAF and 4-NAF. The left-most column is always the fastest method. For the other algorithms, the execution time is given together with the additional time as a percentage of the fastest method.

TABLE 10
Comparison of 24-HBTF, 18-HBTF, and 4-NAF for Weierstrass Curves

Size	24-HBTF	18-HBTF		4-NAF	
192	2.46	2.47	(+0%)	2.54	(+3%)
224	3.21	3.22	(+0%)	3.31	(+3%)
256	3.97	3.99	(+1%)	4.11	(+4%)
384	9.02	9.11	(+1%)	9.30	(+3%)
521	16.90	17.08	(+1%)	17.46	(+3%)

TABLE 11
Comparison of 36-HBTF and 5-NAF for DIK3 Curves

Size	36-HBTF	5-NAF	
192	2.34	2.80	(+20%)
224	3.02	3.58	(+19%)
256	3.64	4.34	(+19%)
384	8.29	9.69	(+17%)
521	16.28	18.76	(+15%)

TABLE 12
Comparison of 36-HBTF and 5-NAF for Weierstrass Curves

Size	36-HBTF	5-NAF	
192	2.48	2.55	(+3%)
224	3.23	3.30	(+2%)
256	3.96	4.06	(+3%)
384	8.99	9.17	(+2%)
521	16.84	17.16	(+2%)

TABLE 13
Comparison of HBTJF, Interleaving 5-NAF, and Interleaving 4-NAF for DIK3 Curves

Size	HBTJF	Inter. 5-NAF		Inter. 4-NAF	
192	3.01	3.32	(+10%)	3.41	(+13%)
224	3.89	4.23	(+9%)	4.34	(+12%)
256	4.68	5.20	(+11%)	5.36	(+15%)
384	10.41	11.53	(+11%)	11.98	(+15%)
521	20.11	22.04	(+10%)	22.99	(+14%)

In Tables 5 and 6, we first compare methods that do not require any precomputation (other than the point P), namely, 6-HBTF, NAF and (left-to-right) double-base chains [14]. For both DIK3 (Table 5) and Weierstrass (Table 6) curves, 6-HBTF is always faster with significant speed ups. Although the number of point addition is generally smaller than for 6-HBTF, the relative poor performance of DB-chains is due to the time spent in the computation of these chains.

In Tables 7 and 8, 18-HBTF and 12-HBTF, which require two and one precomputed points, respectively, are compared to 3-NAF (one point). Significant speed ups are obtained for DIK3 curves, with 18-HBTF slightly faster than 12-HBTF. A small gain is achieved for Weierstrass curves.

TABLE 14
Comparison of HBTJF, Interleaving 5-NAF, and Interleaving 4-NAF for Weierstrass Curves

Size	HBTJF	Inter. 5-NAF		Inter. 4-NAF	
192	3.00	3.03	(+1%)	3.09	(+3%)
224	3.90	3.91	(+0%)	4.02	(+3%)
256	4.81	4.83	(+0%)	4.97	(+3%)
384	10.80	10.84	(+0%)	11.28	(+4%)
521	20.05	20.21	(+1%)	20.92	(+4%)

TABLE 15
Comparison of RHBTJF, JDBC, and JSF for DIK3 Curves

Size	RHBTJF	JDBC		JSF	
192	3.1	3.62	(+17%)	3.69	(+19%)
224	3.95	4.52	(+14%)	4.67	(+18%)
256	4.84	5.57	(+15%)	5.74	(+19%)
384	11.06	12.51	(+13%)	12.95	(+17%)
521	21.49	23.7	(+10%)	25.11	(+17%)

TABLE 16
Comparison of RHBTJF, JSF, and JDBC for Weierstrass Curves

Size	RHBTJF	JSF		JDBC	
192	3.11	3.33	(+7%)	3.46	(+11%)
224	4.04	4.33	(+7%)	4.49	(+11%)
256	4.97	5.36	(+8%)	5.53	(+11%)
384	11.4	12.21	(+7%)	12.42	(+9%)
521	21.41	22.68	(+6%)	23.08	(+8%)

In Tables 9 and 10, 18-HBTF and 24-HBTF, which require two and three precomputed points respectively, are compared to 4-NAF (three points). For DIK3 curves, 18-HBTF is always the fastest method. Our results show a gain of roughly 20 percent compared to 4-NAF. For Weierstrass curves, however, 24-HBTF is only slightly faster than 18-HBTF and the gain compared to 4-NAF is marginal.

Finally, in Tables 11 and 12, we compare 36-HBTF (five precomputed points) to 5-NAF (seven points). On both families of curves, the best results are obtained for 36-HBTF.

Now, we focus our analysis on double-scalar multiplication. In Tables 13 and 14, we compare our hybrid binary-ternary joint form (HBTJF), which requires 14 precomputed points, with interleaving 4-NAF (six points) and interleaving 5-NAF (14 points). Over DIK3 curves, HBTJF is faster than both methods and should be used when the amount of storage can be afforded. For Weierstrass curves, however, it seems that interleaving 4-NAF is a better choice since fewer points need to be stored and the performance is equivalent to that of HBTJF.

Finally, in Tables 15 and 16, we present simulation data for our reduced hybrid binary-ternary joint form, JSF, and joint double-base chains (JDBC) [37]. All three methods only require two precomputations, namely, $P+Q$ and

$P - Q$. Over both tripling-oriented DIK curves and Weierstrass curves, RHBTJF outperforms JSF and JDBC quite significantly.

6 Conclusions

Three novel algorithms have been proposed and thoroughly analyzed. The first one, called w-HBTF is a family of algorithms for single scalar multiplication. It combines the hybrid binary-ternary number system with widely used windowing methods. The other two algorithms, namely, HBTJF and RHBTJF, are for double scalar multiplication.

Which algorithm should be used for the implementation of an elliptic curve protocol depends on several parameters: the amount of memory that is available to store the precomputed points, the size of the finite field, and the type of elliptic curve. For elliptic curves with fast tripling, like tripling-oriented DIK curves, our algorithms are likely to provide significant improvements. Fast tripling algorithms have also been proposed for Weierstrass curves (see [31]). Our experimental results show that, even in that case, our hybrid algorithms are almost always faster than classical w-NAF methods or JSF.

Appendix

Transition Matrix for Hybrid Binary-Ternary Joint Form

The transition matrix is depicted at the bottom of this page.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Part of this work was done during second author's sabbatical leave at Nanyang Technological University, Singapore and during the third author's leave at the University of Calgary under a PIMS-CNRS agreement.

$$
M_D = \begin{pmatrix}
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac19 & 0 & \frac19 & 0 & \frac19 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac14 & 0 & 0 & \frac14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

REFERENCES

[1] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. of Computation*, vol. 48, pp. 203-209, 1987.

[2] V.S. Miller, "Use of Elliptic Curves in Cryptography," *Proc. Conf. Advances in Cryptology (CRYPTO' 85)*, pp. 417-426, 1986.

[3] I.F. Blake, G. Seroussi, and N.P. Smart, *Elliptic Curves in Cryptography*, first ed. Cambridge Univ. Press, 2000.

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

[5] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, July 2005.

[6] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. IT-22, no. 6, pp. 644-654, Nov. 1976.

[7] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, pp. 120-126, 1978.

[8] A.D. Booth, "A Signed Binary Multiplication Technique," *Quarterly J. Mechanics and Applied Math.*, vol. 4, no. 2, pp. 236-240, 1951.

[9] G.W. Reitwiesner, "Binary Arithmetic," *Advances in Computers*, vol. 1, pp. 231-308, Academic Press, 1960.

[10] D.E. Knuth, *The Art of Computer Programming Vol. 2: Seminumerical Algorithms*, Addsion Weslsey Longman Publishing Group, May 1969.

[11] M. Joye and S.-M. Yen, "Optimal Left-to-Right Binary Signed-Digit Exponent Recoding," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 740-748, July 2000.

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

[13] V.S. Dimitrov, L. Imbert, and P.K. Mishra, "Efficient and Secure Elliptic Curve Point Multiplication Using Double-Base Chains," *Proc. Int'l Conf. Theory and Applications of Cryptology and Information Security (ASIACRYPT '05)*, pp. 59-78, 2005.

[14] V.S. Dimitrov, L. Imbert, and P.K. Mishra, "The Double-Base Number System and Its Application to Elliptic Curve Cryptography," *Math. of Computation*, vol. 77, no. 262, pp. 1075-1104, 2008.

[15] C. Doche and L. Imbert, "Extended Double-Base Number System with Applications to Elliptic Curve Cryptography," *Proc. Conf. Progress in Cryptology (INDOCRYPT '06)*, pp. 335-348, Dec. 2006.

[16] F. Morain and J. Olivos, "Speeding Up the Computations on an Elliptic Curve Using Addition-Subtraction Chains," *Theoretical Informatics and Applications*, vol. 24, pp. 531-543, 1990.

[17] H. Cohen, A. Miyaji, and T. Ono, "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates," *Proc. Int'l Conf. Theory and Applications of Cryptology and Information Security (ASIACRYPT '98)*, pp. 51-65, 1998.

[18] M. Ciet and F. Sica, "An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm," *Proc. Conf. Progress in Cryptology (MYCRYPT '05)*, pp. 171-182, 2005.

[19] R. Avanzi, V.S. Dimitrov, C. Doche, and F. Sica, "Extending Scalar Multiplication Using Double Bases," *Proc. Int'l Conf. Theory and Applications of Cryptology and Information Security (ASIACRYPT '06)*, p. 130, 2006.

[20] N. Méloni and M.A. Hasan, "Elliptic Curve Point Scalar Multiplication Combining Yao's Algorithm and Double Bases," *Proc. Workshop Cryptographic Hardware and Embedded Systems*, Sept. 2009.

[21] P. Longa, "Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields," Master's thesis, Univ. of Ottawa, 2007.

[22] P. Longa and C. Gebotys, "Fast Multibase Methods and Other Several Optimizations for Elliptic Curve Scalar Multiplication," *Proc. Conf. Public Key Cryptography (PKC '09)* pp. 443-462, 2009.

[23] J.A. Solinas, "Low-Weight Binary Representations for Pairs of Integers," Research Report CORR 2001-41, Center for Applied Cryptographic Research, Univ. of Waterloo, 2001.

[24] C. Doche, D.R. Kohel, and F. Sica, "Double-Base Number System for Multi-Scalar Multiplications," *Proc. 28th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '09)*, Apr. 2009.

[25] E.G. Straus, "Addition Chains of Vectors (Problem 5125)," *Am. Math. Monthly*, vol. 70, pp. 806-808, 1964.

[26] V.S. Dimitrov and T.V. Cooklev, "Two Algorithms for Modular Exponentiation Based on Nonstandard Arithmetics," *IEICE Trans. Fundamentals of Electronics, Comm. and Computer Science*, vol. E78-A, no. 1, special issue on cryptography and information security, pp. 82-87, Jan. 1995.

[27] J. Adikari, V.S. Dimitrov, and L. Imbert, "Hybrid Binary-Ternary Joint form and Its Application in Elliptic Curve Cryptography" *Proc. 19th IEEE Symp. Computer Arithmetic (ARITH)*, pp. 76-83, June 2009.

[28] A.H. Koblitz, N. Koblitz, and A. Menezes, "Elliptic Curve Cryptography: The Serpentine Course of a Paradigm Shift," to be published in *J. Number Theory*.

[29] C. Doche, T. Icart, and D.R. Kohel, "Efficient Scalar Multiplication by Isogeny Decompositions," *Proc. Conf. Public Key Cryptography (PKC '06)*, pp. 191-206, 2006.

[30] D.J. Bernstein and T. Lange, "Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication," *Finite Fields and Applications, Contemporary Mathematics*, vol. 461, pp. 1-19, Am. Math. Soc., 2008.

[31] D.J. Bernstein and T. Lange, "Explicit-Formulas Database," Joint Work by D.J. Bernstein and T. Lange, Building on Work by Many Authors. http://www.hyperelliptic.org/EFD/, 2010.

[32] 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.

[33] V.S. Dimitrov, G.A. Jullien, and W.C. Miller, "Theory and Applications of the Double-Base Number System," *IEEE Trans. Computers*, vol. 48, no. 10, pp. 1098-1106, Oct. 1999.

[34] V.S. Dimitrov and G.A. Jullien, "Loading the Bases: A New Number Representation with Applications," *IEEE Circuits and Systems Magazine*, vol. 3, no. 2, pp. 6-23, Nov. 2003.

[35] V.S. Dimitrov, G.A. Jullien, and W.C. Miller, "An Algorithm for Modular Exponentiation," *Information Processing Letters*, vol. 66, no. 3, pp. 155-159, 1998.

[36] V.S. Dimitrov, G.A. Jullien, and W.C. Miller, "Theory and Applications for a Double-Base Number System," *Proc. 13th Symp. Computer Arithmetic (ARITH)*, p. 44, July 1997.

[37] C. Doche and L. Habsieger, "A Tree-Based Approach for Computing Double-Base Chains," *Proc. 13th Australasian Conf. Information Security and Privacy (ACISP '08)*, pp. 433-446, 2008.

[38] D.J. Bernstein, P. Birkner, T. Lange, and C. Peters, "Optimizing Double-Base Elliptic-Curve Single-Scalar Multiplication," *Proc. Conf. Progress in Cryptology (INDOCRYPT '07)*, pp. 167-182, 2007.

[39] Nat'l Inst. of Standards and Technology, *FIPS 186-2, Digital Signature Standard*, Fed. Information Processing Standards Publication, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, 2000.

[40] C. Doche, "Tripling Oriented DIK Curve Software Implementations," Personal Communication, Sept. 2008.

[41] K. Okeya and K. Sakurai, "Use of Montgomery Trick in Precomputation of Multi-Scalar Multiplication in Elliptic Curve Cryptosystems," *Trans. Fundamentals of Electronics, Comm. and Computer Science*, vol. E86-A, no. 1, pp. 98-112, 2003.

[42] H. Cohen, *A Course in Computational Algebraic Number Theory*, third ed. Springer, Sept. 1993.

[43] Free Software Foundation, "GMP, Arithmetic without Limitations," http://www.gmplib.org/, Apr. 2009.

Jithra Adikari (S'07) received the BSc degree in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, and the MSc degree in information and communication systems security from the Royal Institute of Technology (KTH), Stockholm, Sweden. He is currently working toward the PhD degree at the University of Calgary, AB, Canada. He has been with the Advanced Technology Information Processing Systems (ATIPS) Group, University of Calgary, since January 2007. His research interests include efficient software/hardware implementations of elliptic curve cryptographic algorithms, computer arithmetic, and digital signal processing. He is a member of the IEEE.

Vassil S. Dimitrov received the PhD degree in mathematics from the Mathematical Institute of the Bulgarian Academy of Sciences in 1995. He holds research scientist positions at the University of Windsor, Canada, Helsinki University of Technology, Finland, and Nanyang Technological University, Singapore. Since 2001, he has been an associate professor at the Department of Electrical and Computer Engineering, University of Calgary, Canada. His research interests include implementation of cryptographic protocols, computational complexity problems, the use of number-theoretic techniques in digital signal processing, image compression, and related topics.

Laurent Imbert received the MSc and PhD degrees in computer science from the Université Aixin-Marseille in 1997 and 2000, respectively. Since October 2001, he has been a tenured researcher at the Centre National de la Recherche Scientifique (CNRS), working in the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), France. He is the head of the LIRMM's Security Group. His research interests cover both theoretical and practical aspects of arithmetic and cryptography.