Improved Algorithms for Efficient Arithmetic on Elliptic Curves Using Fast Endomorphisms

Mathieu Ciet[1⋆], Tanja Lange[2], Francesco Sica[1⋆], and Jean-Jacques Quisquater[1]

[1] UCL Crypto Group
Place du Levant, 3. B-1348 Louvain-la-Neuve. Belgium
{ciet, sica, jjq}@dice.ucl.ac.be – http://www.dice.ucl.ac.be/crypto/
[2] Institute for Information Security and Cryptology (ITSC)
Ruhr-Universität Bochum, Universitätsstraße 150, D-44780 Bochum, Germany
lange@itsc.ruhr-uni-bochum.de– http://www.ruhr-uni-bochum.de/itsc/

Abstract. In most algorithms involving elliptic curves, the most expensive part consists in computing multiples of points. This paper investigates how to extend the τ-adic expansion from Koblitz curves to a larger class of curves defined over a prime field having an efficiently-computable endomorphism ϕ in order to perform an efficient point multiplication with efficiency similar to Solinas' approach presented at CRYPTO '97. Furthermore, many elliptic curve cryptosystems require the computation of k_0P+k_1Q. Following the work of Solinas on the Joint Sparse Form, we introduce the notion of ϕ-Joint Sparse Form which combines the advantages of a ϕ-expansion with the additional speedup of the Joint Sparse Form. We also present an efficient algorithm to obtain the ϕ-Joint Sparse Form. Then, the double exponentiation can be done using the ϕ endomorphism instead of doubling, resulting in an average of l applications of ϕ and $l/2$ additions, where l is the size of the k_i's. This results in an important speed-up when the computation of ϕ is particularly effective, as in the case of Koblitz curves.

Keywords. Elliptic curves, fast endomorphisms, Joint Sparse Form.

1 Introduction

Let $\mathbb{F}_q$, with $q = p^\ell$, p prime, be a finite field. In cryptography, one is mainly interested in the following two cases: $q = 2^\ell$ (binary fields) or $q = p > 3$ (prime fields). Let E be an elliptic curve defined over $\mathbb{F}_q$ and $P \in E(\mathbb{F}_q)$ a point of

⋆ The work described in this paper has been supported [in part] by the Commission of the European Communities through the IST Programme under Contract IST-1999-12324, http://www.cryptonessie.org/. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. The views expressed are those of the authors and do not represent an official view/position of the NESSIE project (as a whole).

large prime order n (typically $n > q/5$). In elliptic curve cryptography, it is essential to be able to compute quickly a multiple kP for any $k \in [1, n-1]$. A few methods use fast computable endomorphisms ϕ [9, 11, 12, 14, 15, 17, 19–21]. For binary Koblitz curves [11, 12, 21], it is standard to use as ϕ the Frobenius endomorphism over $\mathbb{F}_2$ (often denoted τ). One then gets a decomposition

$$kP = k_0 P + k_1 \phi(P) + \cdots + k_m \phi^m(P) \ , \tag{1}$$

with the $k_i = 0, \pm 1$, similar to the signed binary decomposition of k. Using the fact that $\phi^\ell(P) = P$, one can take $m = \lceil \log_2 n \rceil$.

Over prime fields, one uses an effective endomorphism ϕ such that its minimal polynomial $X^2 + rX + s$ has small coefficients: this is the method of Gallant-Lambert-Vanstone (GLV for short). The GLV method [9, 17, 19] therefore works on those elliptic curves over $\mathbb{F}_p$ with endomorphism ring having small discriminant. The substance of the GLV method is to decompose kP as

$$kP = k_0 P + k_1 \phi(P), \quad \text{with } \max\{|k_0|, |k_1|\} = O(\sqrt{n}) \tag{2}$$

and then compute $k_0 P$ and $k_1 \phi(P)$ "simultaneously" (this is true if one can parallelize the architecture, otherwise, some speedup can still be obtained by Solinas' Joint Sparse Form (JSF) [23]). In practice, the constant in the $O(\sqrt{n})$ estimate is small (around $\sqrt{4s - r^2}/4$, see [19]) and in the examples can even be taken to be 1.

Our first contribution is to show that the GLV algorithm is just the first ingredient to get a generalized base-ϕ expansion leading to the same kind of decomposition as (1), with $k_i \in \mathcal{R} = \{-u, \ldots, u\}$ and u small – in the examples we present even $u = 1$. To use such an expansion one applies Horner's rule $kP = \phi(\phi(\cdots \phi(k_m P) + k_{m-1} P) + \cdots + k_1 P) + k_0 P$. If u is small one can easily precompute all $u_i P$, $0 \leq u_i \leq u$. Then this computation reduces to m applications of ϕ and for each non-zero coefficient k_i one table look-up and one addition.

The efficiency of this method relies heavily on the ratio of the costs for curve doublings and the operation of ϕ. We show that for the examples these expansions lead to faster scalar multiplication than the binary method and compare it to the GLV method.

Our second contribution is the development of a fast algorithm to perform double exponentiations à la Solinas, when a fast endomorphism is available. Indeed, there are several occasions where one computes $k_0 P + k_1 Q$, e. g. to check a signature or when applying the GLV method (then $Q = \phi(P)$). The JSF is a standard tool to speed up this computation. It decomposes the multipliers in base 2 and achieves a joint density of 1/2.

If the curve is such that there exists an efficiently computable endomorphism ϕ, we can combine the speed-up gained by using a ϕ expansion like (1) together with the JSF in a nontrivial fashion to obtain what we call the ϕ-JSF. If the characteristic polynomial of ϕ is $P(X) = X^2 \pm X + 2$ or $P(X) = X^2 + 2$ (these polynomials are the main cases of interest as they occur in the applications) we obtain the same density of 1/2 and a similar length of the expansion.

Applications are to further speed up the scalar multiplication on Koblitz curves and to check signatures on them more efficiently (Koblitz curves are suggested in the NIST standard [16]). If ϕ is an endomorphism of a prime field curve then the ϕ-JSF can similarly be applied and can give better performance than the GLV method combined with the original JSF. For simplicity, we assume minimal precomputation. In general, allowing more precomputations additionally speeds up the scalar multiplication, see [2] for an overview of the applications we consider here.

2 Basic Notations and Preliminaries

Here we briefly introduce elliptic curves and review techniques for binary expansions that are used later on. An elliptic curve over a finite field $\mathbb{F}_q$ can be given by a Weierstraß equation

$$E : y^2 + (a_1x + a_3)y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in \mathbb{F}_q.$$

The group one uses consists of the affine points $(x, y) \in \mathbb{F}_q^2$ satisfying the equation along with a point $\mathcal{O}$ at infinity. Depending on the implementation environment different systems of coordinates might be advantageous; the following two systems avoid inversions in the group operations. In projective coordinates a point is represented as $(X : Y : Z)$ with $x = X/Z, y = Y/Z$, in Jacobian coordinates $(X : Y : Z)$ stands for the affine point $(X/Z^2, Y/Z^3)$. In affine coordinates an addition of distinct points takes 1 inversion, 2 multiplications (M) and 1 squaring (S) whereas a doubling takes one more squaring, in projective coordinates an addition is computed using 12M and 2S and a doubling in 7M and 5S. Jacobian coordinates lead to 12M and 4S for an addition and 4M and 6S for a doubling. For more details we refer to [6]. For special choices of the coefficients a_i fewer operations are needed.

We now turn our attention to (signed) binary expansions. By *density* of an expansion we mean the number of nonzero coefficients (Hamming weight) divided by the length of the expansion. The Non Adjacent Form (NAF) of an integer k is a signed binary expansion $k = \sum_i k_i 2^i$ with $k_i \in \{0, \pm 1\}$ and $k_i k_{i+1} = 0$ (see e.g. [13, 18, 21] for algorithms to compute it). The average density of a NAF expansion is approximately 1/3, for an ordinary binary expansion it is 1/2. Shamir [7] rediscovered an efficient trick, originally due to Straus [24], to speed up the evaluation of $k_0P + k_1Q$ (see [3]; also [10] for a survey of exponentiation algorithms). A naive way for this double scalar multiplication is to compute both powers separately needing $2l$ doublings and l additions on average if k_0 and k_1 have length l. Shamir's proposal resulted in two methods, one called *simple Straus-Shamir method* and the other *fast Straus-Shamir method.* The last one requires the precomputation of $P + Q$ and $k_0P + k_1Q$ is evaluated with l doublings and $3l/4$ additions on average.

In [23] Solinas extended the Straus-Shamir method to the case of signed binary expansions, which is useful for groups where negating is cheap, coming up with the Joint Sparse Form. This is especially useful for jacobians of elliptic and hyperelliptic curves, where point inversion is virtually free. He gave an efficient algorithm to compute the JSF and also proves some properties of this decomposition. Let us briefly recall the axioms defining this decomposition (cf. [23] for notation):

(JSF 1) Of any three consecutive columns at least one is a zero column.
(JSF 2) It is never the case that $u_{i,j+1}u_{i,j} = -1$.
(JSF 3) If $u_{i,j+1}u_{i,j} \neq 0$ then $u_{1-i,j+1} = \pm 1$ and $u_{1-i,j} = 0$.

Example 1. Let $k_0 = 403$ and $k_1 = 334$, the NAF expansions [13] of k_0 and k_1 are given (in big endian notation) on the left, while the JSF is on the right.

$$k_0 = \langle\ 1\ 0\ \text{-}1\ 0\ 0\ 1\ 0\ 1\ 0\ \text{-}1\ \rangle = \langle\ 1\ 0\ \text{-}1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ \rangle$$
$$k_1 = \langle\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ \text{-}1\ 0\ \rangle = \langle\ 1\ 0\ \text{-}1\ \text{-}1\ 0\ 1\ 0\ 0\ \text{-}1\ 0\ \rangle$$

Define the joint Hamming weight of any joint binary expansion of two integers written down as in the example to be the number of nonzero columns. The joint Hamming weight gives the number of additions $\pm P$, $\pm Q$, $\pm(P+Q)$ or $\pm(P-Q)$ to perform during the course of the joint double and add algorithm to compute $k_0P + k_1Q$. Since the JSF is a generalization of the *fast* Straus-Shamir method, it is supposed that P, Q, $P+Q$ and $P-Q$ have been precomputed and stored. Hence to make the computation less expensive, it is vital to get the lowest possible joint Hamming weight. In the example, the joint NAF decomposition has joint Hamming weight 8, whereas the JSF lowers it to 6.
The JSF has many nice properties, which we recapitulate here.

Theorem 1 (from [23]). *The Joint Sparse Form of any two integers exists and is unique. It has minimal joint Hamming weight among all joint signed binary expansions. If k_0 and k_1 have maximal length l, then the joint double and add algorithm computes $k_0P + k_1Q$ from the JSF with an average of l doublings and $l/2$ additions of either $\pm P$, $\pm Q$, $\pm(P+Q)$ or $\pm(P-Q)$.*

If one cannot afford to store and precompute 3 points then the best way is to take k_0, k_1 both in NAF representation. The joint density is 5/9 on average. Of this 5/9 proportion of non-zero columns, 1/9 has two non-zero entries and 4/9 exactly one zero entry. Without precomputation, in the former case the joint double and add algorithm has to perform two additions at that step while in the latter one addition is needed. For k_0, k_1 both of length l, this amounts to $2l/3$ additions and l doublings. Hence, compared to the naive way, the number of additions remains unchanged but the number of doublings is halved.

3 Key Decomposition: ϕ-Expansions

The aim of this section is to describe how to obtain a decomposition to the base of ϕ. Recall that ϕ is an endomorphism of the curve with $X^2 + rX + s$ as minimal polynomial. We assume $s > 1$. Given $z \in \mathbb{Z}[\phi]$, we want to decompose it as

$$z = k_0 + k_1\phi + \cdots + k_m\phi^m \quad \text{with } k_i \in \mathcal{R}\ . \tag{3}$$

We find the coefficients $k_0, \ldots, k_m$ inductively. The following lemma is needed.

Lemma 1 (Lemma 5.3 of [12]). *Let $a, b \in \mathbb{Z}$, then ϕ divides $a + b\phi$ in $\mathbb{Z}[\phi]$ if and only if s divides a.*

This implies that a choice of $\mathcal{R}$ as a complete set of residues modulo s is necessary and sufficient to guarantee the existence and uniqueness of k_0. As taking the negative of a point is for free, we choose a set of remainders symmetric w. r. t. zero. When s is odd, a complete choice is $\mathcal{R} = \{-(s-1)/2, \ldots, (s-1)/2\}$. In even characteristic we include both $-s/2$ and $s/2$ without enlarging the number of precomputations. Thus when s is even we take $\mathcal{R} = \{-s/2, \ldots, s/2\}$. From now on we stick to this choice of $\mathcal{R}$.

Let $z = z_0 + z_1\phi \in \mathbb{Z}[\phi]$. To decompose z we put $k_0 \equiv z_0 \bmod s$. Then using the minimal polynomial $X^2 + rX + s$ of ϕ, i. e. $s = -r\phi - \phi^2$, we get

$$z = k_0 + \frac{z_0 - k_0}{s}s + z_1\phi = k_0 + \phi\left(\left(\frac{k_0 - z_0}{s}\,r + z_1\right) + \frac{k_0 - z_0}{s}\,\phi\right).$$

We then replace z by $(z - k_0)/\phi$ and find k_1, then replace z by $(z - k_1)/\phi$ and compute the coefficients iteratively. The main question is now to show that this process stops after finitely many steps and to bound the length of these expansions.

Theorem 2. *Let $s > 1$. Then $z = z_0 + z_1\phi \in \mathbb{Z}[\phi]$ can be expanded as* (3) *with $m \leq \lceil 2\log_s 2\sqrt{z_0^2 - rz_0z_1 + sz_1^2}\rceil + 3$ except when $(r, s) = (\pm 2, 2), (\pm 3, 3), (\pm 4, 5)$ or $(\pm 5, 7)$. In these cases one has to allow $k_{m-1} = \pm\lceil (s+1)/2\rceil$.*

Proof. The proof follows the same lines as the corresponding proofs in [14] and [20]. The fact that ϕ is not the Frobenius is not important at this stage. What really matters is that the complex norm of ϕ is s. The additional cases of small s which cannot occur as characteristic polynomial of the Frobenius endomorphism (e. g. for s not a prime power) have been checked by hand. □

Note that for $z = k \in \mathbb{Z}$ an integer multiplier, these theorems give a decomposition with length approximately $2\log_s |k| \approx 2\log_s n$ for values of k used in cryptography. This is twice as long as a s-ary expansion. If ϕ is the Frobenius one can shorten the length in reducing k modulo $\phi^\ell - 1$ before expanding (see [12, 21]). Closing up this gap for prime field curves in the fashion of previous authors is therefore necessary to gain advantage of this decomposition.

However, it is clear from previous research (see the use of λ-Euclidean rings in [17, 20]) that in fact one can cut down the length of the decomposition of the multiplier k by replacing it by $z_0 + z_1\phi, z_0, z_1 \in \mathbb{Z}$, such that $kP = z_0P + z_1\phi(P)$ and $\max(|z_0|, |z_1|) = O(\sqrt{n})$. But this is precisely the meaning of the Gallant-Lambert-Vanstone (GLV) method for these curves. Then a direct application of [19, Theorem 1] gives the following.

Theorem 3. *Let P be a point of large prime order n on an elliptic curve and ϕ a non trivial endomorphism such that $\phi^2 + r\phi + s = 0$. Then, for an arbitrary $1 \leq k \leq n$ the above algorithm coupled with a GLV reduction gives a decomposition* (1) *where $k_i \in \mathcal{R}$ (with the exceptions listed in Theorem 2) and $m \leq \lceil 2\log_s 2\sqrt{1 + |r| + s} + \log_s n\rceil + 3$.*

In the case where n is the norm of some element of $\mathbb{Z}[\phi]$ (which is true if $\mathbb{Z}[\phi]$ is integrally closed and principal), applying [19, Theorem 3] we can replace $\sqrt{1+|r|+s}$ by a smaller value. However, for practical purposes the previous theorem is clearly sufficient since now – up to a few constant bits – we can achieve a decomposition of length $\log_s n$.

For $s = 1$ a ϕ-expansion must have exponential length. It is not hard to show in this case that if in (3) we have $|k_i| \leq u$, then one gets that $m > \sqrt{n}/(2u)$ so that u must be very large in order to get $m = O(\log n)$. Therefore, we cannot apply such a decomposition efficiently.

In the well-studied cases, where ϕ is the Frobenius endomorphism, these expansions lead to a large speed-up. Application of ϕ then corresponds to p-th powering. If the field elements are represented with respect to a normal basis, the application of ϕ is for free as it is performed via a cyclic shift of the components. For a polynomial basis these costs cannot be neglected but are significantly less than a group operation, independently of the coordinate system we use to represent the points.

For other endomorphisms we quote here two examples appearing already in [5, 9, 17]. Note that in these examples, $\mathbb{Z}[\phi]$ is the maximal order and it is principal, and that $s = 2$. Using complex multiplication one can construct further examples. For larger s the expansions get shorter. However, in light of what follows these examples with $s = 2$ are of special interest. Here, we compare the number of operations to compute the endomorphism to those needed for point doublings in the same set of coordinates. We choose projective coordinates as then the number of operations needed to compute ϕ is minimal and the additions are cheaper than in Jacobian coordinates.

Example 2. Let $p > 3$ be a prime such that -7 is a quadratic residue modulo p. Define an elliptic curve E_3 over $\mathbb{F}_p$ by $y^2 = x^3 - 3x^2/4 - 2x - 1$. If $\xi = (1+\sqrt{-7})/2$ and $a = (\xi - 3)/4$, then the map ϕ defined in the affine plane by

$$\phi(x, y) = \left(\frac{x^2 - \xi}{\xi^2(x - a)}, \frac{y(x^2 - 2ax + \xi)}{\xi^3(x - a)^2} \right) \quad ,$$

is an endomorphism of E_3 defined over $\mathbb{F}_p$ with $\mathbb{Z}[\phi] = \mathbb{Z}[\frac{1+\sqrt{-7}}{2}]$. Moreover ϕ satisfies the equation $\phi^2 - \phi + 2 = 0$. In affine coordinates, the formulæ given previously are clearly more expensive, as already noticed in [9, 17], than doubling [1, 4]. However, in projective coordinates, $\phi(X, Y, Z)$ is given by $\phi(X, Y, Z) = (EF, Y(A - 2XD + C), F^2B)$ with $A = X^2$, $B = \xi Z$, $C = BZ$, $D = aZ$, $E = A - C$ and $F = (X - D)\xi$.
Then, given a point $P = (X, Y, Z)$ its image by ϕ is computed with 8 multiplications and 2 squarings compared to 7 multiplications and 5 squarings for point doubling.

Example 3. Let $p > 3$ be a prime such that -2 is a quadratic residue modulo p. Define an elliptic curve E_4 over $\mathbb{F}_p$ by $y^2 = 4x^3 - 30x - 28$. The map ϕ defined in the affine plane by

$$\phi(x,y) = \left(-\frac{2x^2+4x+9}{4(x+2)}, -\frac{2x^2+8x-1}{4\sqrt{-2}(x+2)^2}y\right)$$

is an endomorphism of E_4 defined over $\mathbb{F}_p$ with $\mathbb{Z}[\phi] = \mathbb{Z}[\sqrt{-2}]$. Moreover ϕ satisfies the equation $\phi^2 + 2 = 0$. As in the previous example, the endomorphism formulæ are more expensive than those for doubling in affine coordinates. However, in projective coordinates the endomorphism can be computed as $\phi(X, Y, Z) = (D(2A + 4B + 9Z^2), (2A + 8B - Z^2)\,Y, -4DCZ)$ with $A = X^2$, $B = XZ$, $C = X + 2Z$ and $D = \sqrt{-2}\,C$.
Therefore, this endomorphism is significantly faster than a doubling since given a point P in projective coordinates, $\phi(P)$ can be computed with only 6 multiplications and 2 squarings [3].

Density. We now show that we can lower the density of the ϕ-expansion, for the expansions of the examples from the obvious 1/2 to 1/3. For applications with larger s similar considerations hold but the effects are not that dramatic, however, the length is shorter as a compensation. In [21] Solinas considers expansions to the base of the Frobenius τ for Koblitz curves. The characteristic polynomial of τ for the curves $y + xy = x^3 + ax^2 + 1$ over $\mathbb{F}_2$ is given by $X^2 + (-1)^a X + 2$. He introduces a τ-Non Adjacent Form (τ-NAF) and states algorithms to compute kP as $kP = \sum_i k_i \tau^i(P)$ with $k_i = 0, \pm 1$ and $k_i k_{i+1} = 0$. Such an expansion has an average density of 1/3.

This characteristic polynomial coincides with the one of ϕ in Example 2. Thus also in this case we can compute a ϕ-NAF expansion of density 1/3 by exactly the same algorithm. In the second example we have $\phi^2 = -2$. To obtain a lower density of the expansion we impose a further condition on the k_i in the expansion: given $z_0 + z_1\phi$, for $z_0 \equiv 0 \bmod 2$ choose $k_0 = 0$ as before. Otherwise, put $k_0 \equiv z_0$ mods 4, where mods 4 means that one chooses ± 1 as representatives modulo 4. This gives $2 \mid (z_0 - k_0)/2$, which sets to zero the next but one coefficient; and thus there is at least one zero coefficient for each nonzero one (in this case $k_i k_{i+2} = 0$), again leading to a density of 1/3. In practice, in this case the ϕ-NAF expansion is obtained from signed binary expansions of z_0 and z_1. We refer to such expansions as ϕ-NAFs.

Complexity and comparison with signed binary method. In the examples, the computation of kP using ϕ-expansions amounts approximately to $\log_2 n$ applications of ϕ and $\log_2 n/3$ additions. The expansions are of the same length and density as the binary expansion but the doublings are replaced by cheaper applications of ϕ. For both of these examples we thus obtain that computing scalar multiples using a ϕ-expansion is more efficient than via the binary method as $\phi(P)$ needs less than a doubling. This holds as well if the binary method uses Jacobian coordinates.

[3] We count the multiplication of a number by $\sqrt{-2}$ as one multiplication.

Comparison with GLV method. As already mentioned, the GLV method leads to a decomposition $k = k_0 + k_1\phi$. The binary length of k_i is $\log_2 n/2$. Taking both k_i in binary NAF form, GLV needs $\log_2 n/3$ additions and $\log_2 n/2$ doublings if $\phi(P)$ is precomputed. We remark that our new method works without precomputations and then needs the same number of additions but replaces the doublings by two times the number of applications of ϕ. Unfortunately, in our examples two applications of ϕ need more operations than a doubling.

Following ideas similar to Solinas [21], one can adjust the computation of the ϕ-adic expansion to allow an efficient use of windowing techniques. Thus allowing one precomputation as in the initial GLV method the number of additions in our method reduces to $\log_2 n/4$.

If we use the JSF of k_0, k_1 the number of additions drops down to $\log_2 n/4$ in GLV. Using the JSF however implies that one precomputes 3 points. Using 3 precomputed points with our method as well, the density of the ϕ-expansion reduces to 1/5. Applying this to the above examples we notice that the ϕ-expansion is slightly slower than the GLV method no matter if GLV uses projective or Jacobian coordinates. However, the next section provides an improvement.

4 On the Joint Sparse Form

We now aim at combining both methods – the ϕ-expansion with the JSF. If we need to compute $k_0P + k_1Q$, as for example in ECDSA [8], on a curve with efficient endomorphisms we can decompose k_0 and k_1 in base ϕ, but so far the Joint Sparse Form can only be used with a binary expansion. Given the k_i in ϕ-NAF this leads to $2\log_2 n/3$ additions and $\log_2 n$ applications of ϕ without precomputations. Using the 2 precomputed values $P \pm Q$ the number of additions drops down to $5\log_2 n/9$.

In the same spirit as the work of Solinas [23] we introduce the notion of ϕ-Joint Sparse Form (ϕ-JSF), which allows an application of the fast Straus-Shamir method to ϕ-adic expansions of the scalars k_i.

In the following, we denote by ϕ any endomorphism having $X^2 - \epsilon X + 2$ as characteristic polynomial, with $\epsilon = \pm 1$ (for instance the Frobenius endomorphism on binary Koblitz curves or ϕ from Example 2). The correct translation of Solinas' notion of JSF is as follows.

Definition 1 (ϕ-Joint Sparse Form). *A joint expansion with coefficients $0, \pm 1$ is in ϕ-Joint Sparse Form (ϕ-JSF) if it satisfies the following properties:*
(ϕ-JSF 1) *Among three consecutive columns at least one is a double zero.*
(ϕ-JSF 2) *It is never the case that $u_{i,j+1}\, u_{i,j} = \epsilon$.*
(ϕ-JSF 3) *If $u_{i,j+1}\, u_{i,j} \neq 0$ then $u_{1-i,j+1} = \pm 1$ and $u_{1-i,j} = 0$.*

Example 4. On the left we give the joint NAF expansion of k_0 and k_1, on the right the ϕ-JSF ($\epsilon = 1$).

$$\begin{aligned} k_0 &= \langle\, \text{-1 0 -1 0 -1 0 1 0 1} \,\rangle = \langle\, \text{-1 0 0 -1 1 0 0 1 -1} \,\rangle \\ k_1 &= \langle\, \text{0 -1 0 -1 0 0 0 1 0} \,\rangle = \langle\, \text{0 -1 0 -1 0 0 0 1 0} \,\rangle \end{aligned}$$

The ϕ-Joint Sparse Form satisfies properties analogous to the properties of the binary Joint Sparse Form.

Theorem 4. *The ϕ-Joint Sparse Form of any two elements k_0 and k_1 of $\mathbb{Z}[\phi]$ exists and is unique. If k_0 and k_1 have maximal length l when written in ϕ-NAF expansion, then the joint ϕ and add algorithm computes $k_0P + k_1Q$ from the ϕ-JSF with an average of $l+3$ applications of ϕ and $(l+3)/2$ additions of either $\pm P$, $\pm Q$, $\pm(P+Q)$ or $\pm(P-Q)$.*

Proof. The proof will appear in the full version of the paper. It is similar to Solinas' proof, cf. [23]. □

Unfortunately, the minimality of the JSF does not carry over to the ϕ-JSF, since the ϕ-JSF of $(\langle 1, 0, -1\rangle, \langle 0, \epsilon, 0\rangle)$ is $(\langle -\epsilon, 0, -\epsilon, 0, -\epsilon, 1\rangle, \langle 0, 0, 0, 0, \epsilon, 0\rangle)$. However, for large l, the ϕ-JSF has joint Hamming weight differing from the minimum joint Hamming weight at most by a small constant. We now give an algorithm similar to Solinas' Algorithm 2 to produce the ϕ-JSF of k_0 and k_1, assuming they are already written in some ϕ-adic expansion.

Algorithm 1

`Input:` $k_0 = \langle k_{0,m-1}, \ldots, k_{0,0}\rangle$, $k_1 = \langle k_{1,m-1}, \ldots, k_{1,0}\rangle$ in ϕ expansion, put $k_{i,j} = 0$ for $j \geq m$
`Output:` ϕ-JSF of k_0 and k_1

---

1. **initialize:**
   $j \leftarrow 0$
   `For` i `from` 0 `to` 1 `do`
      `Set` $d_{i,0} \leftarrow 0, d_{i,1} \leftarrow 0$
      $a_i \leftarrow k_{i,0}$, $b_i \leftarrow k_{i,1}$, $c_i \leftarrow k_{i,2}$
   `Next` i
2. **main loop:**
   `while` $m - j + |d_{0,0}| + |d_{0,1}| + |d_{1,0}| + |d_{1,1}| > 0$ `do`
   (a) **choose coefficient:**
      `For` i `from` 0 `to` 1 `do`
         `If` $d_{i,0} \equiv a_i \bmod 2$
         `then set` $u \leftarrow 0$
         `else`
         `Set` $u \leftarrow d_{i,0} + a_i + \epsilon 2(d_{i,1} + b_i) \text{ mods } 4^\dagger$
         `If` $d_{i,0} + a_i - \epsilon 2(d_{i,1} + b_i) - 4c_i \equiv \pm 3 \bmod 8$
         `and` $d_{1-i,0} + a_{1-i} + 2(d_{1-i,1} + b_{1-i}) \equiv 2 \bmod 4$
         `then set` $u \leftarrow -u$
         `Set` $u_{i,j} \leftarrow u$
      `Next` i

† Here the notation mods 4 means that one chooses ± 1 as representatives modulo 4 (for an odd number).

(b) **setup for next step:**
`For` i `from` 0 `to` 1 `do`
`Set` $d_{i,0} \leftarrow \epsilon(d_{i,0} + a_i - u_{i,j})/2 + d_{i,1}$
$d_{i,1} \leftarrow \epsilon(d_{i,1} - d_{i,0})$
$a_i \leftarrow b_i$, $b_i \leftarrow c_i$, $c_i \leftarrow k_{i,j+3}$
`Next` i
(c) `Next` j

Explanation. This algorithm is heavily inspired from Solinas' Algorithm 2. There are two main differences. The first one is that there are two carries for each line instead of only one. To draw a parallel with Solinas' algorithm, we introduce the quantity (complex carry) $d_i = d_{i,1}\phi + d_{i,0}$. Then updating the complex carry from step j to step $j+1$ (first two lines of **setup for next step**) is equivalent to replacing these lines with

$$d_i \leftarrow (d_i + a_i - u_{i,j})/\phi.$$

The second one lies in imposing the right condition modulo 8. Namely the condition mod ϕ^3 now reads (by property **(ϕ-JSF 2)**)

$$d_i + \phi^2 c_i + \phi b_i + a_i \equiv \pm(\phi - \epsilon) \bmod \phi^3$$

and this is equivalent to $d_{i,0}+a_i-\epsilon 2(d_{i,1}+b_i)-4c_i \equiv \pm 3 \bmod 8$. Other conditions modulo 2 and 4 are translations of similar congruences mod ϕ and ϕ^2.

When viewing the algorithm as using congruences modulo powers of ϕ and leaving the complex carry without splitting it into $d_{i,1}$ and $d_{i,0}$, one sees the full appearance of the Solinas algorithm and using Solinas' method it is straightforward to check that the above algorithm actually gives the ϕ-JSF of its inputs.

Remark 1. Note that the coefficients $k_{i,j}$ need not necessarily take on values from $\{0, \pm 1\}$. The algorithm works just as well on larger coefficients $k_{i,j}$ (the carries are then unbounded). Thus we need not compute a ϕ-expansion with coefficients in $\{0, \pm 1\}$ prior to applying Algorithm 1. Thus e.g. for signature verification we can apply it directly on the outputs of either the GLV method or the $k_{i,0} + k_{i,1}\phi \equiv k_i \bmod \phi^\ell - 1$ (which is the output of Solinas' algorithm to reduce the length) in the case of Koblitz curves via $k_i = \langle k_{i,1}, k_{i,0}\rangle$.

Remark 2. The case $\phi^2 = -2$ (see Example 3) works in an even simpler way with the modifications mentioned in the previous section.

5 Applications and Comparison

Fix the setting that each multiplier has length l and assume that l is even (to shorten the explanations); all values hold approximately. There are two main cases where the ϕ-JSF can be applied efficiently. One is obviously in signature checking. If one point is going to be reused very often such that a costly precomputation is acceptable, precompute $\phi^{l/2+1}P$. Then one can first compute a ϕ-expansion and then split it in halves as $\sum_{i=0}^{l/2} k_i\phi^i + \phi^{l/2+1}\sum_{i=0}^{l/2} k_{i+l/2}\phi^i$.

1. ϕ is the Frobenius. To check a signature $k_0P + k_1Q$ the ϕ-JSF needs l applications of ϕ and $l/2$ additions. Using ϕ-NAFs of the k_i with the same precomputations and the Straus-Shamir method leads to $5l/9$ additions and again l applications of ϕ (this is just two squarings that are free if optimal normal bases are used). Thus the ϕ-JSF wins over a joint ϕ-NAF. Also, because ϕ is very fast compared to doubling, the ϕ-JSF is much faster than the binary JSF of k_0, k_1 (remember that one first reduces the size of k_i using the trick that a power of Frobenius acts trivially on P and Q).
2. ϕ is the Frobenius. Let $\phi^{l/2+1}P$ be precomputed. To compute kP we first compute the ϕ-adic expansion of k as in [22], then split it and apply the ϕ-JSF on both halves. This needs $l/2$ times ϕ and $l/4$ additions with 3 precomputations. The ordinary method needs l times ϕ and $l/5$ additions for 3 precomputations. ϕ-JSF is faster if an addition takes no more than 10 times ϕ which holds for polynomial basis and all coordinate systems. (Assuming that a multiplication takes at most 3 squarings, in affine coordinates this holds if one inversion takes less than 13 squarings – but otherwise affine coordinates would not be applied.)
3. ϕ is an endomorphism of the examples. If we have precomputed $\phi^{l/2+1}P$ we need $l/2$ times ϕ and $l/4$ additions with 3 precomputations. GLV also uses 3 precomputations and needs $l/2$ doublings and $l/4$ additions. As doublings are more expensive, the ϕ-JSF is faster.
4. ϕ is an endomorphism of the examples and we check signatures. Let the GLV method produce $k_i = k_{i,0} + k_{i,1}\phi$. The input to Algorithm 1 are $k_i = \langle k_{i,1}, k_{i,0}\rangle$. This results in l times ϕ and $l/2$ additions. Taking a binary expansion of k_i and three precomputations, signature verification needs l doublings and $l/2$ additions – thus more than ϕ-JSF.
   In this case the GLV method has to deal with a quadruple multiplication. Using 6 precomputations (grouping together two binary expansions) they need $l/2$ additions and $l/2$ doublings. Thus the ϕ-JSF is advantageous if either ϕ takes less than half a doubling or if the space is more restricted.
   Likewise one can compare this to a quadruple ϕ-expansion using the trick above which results in $l/2$ additions and $l/2$ applications of ϕ, using 6 precomputations. Thus, if one can afford these expensive precomputations and the storage, the ϕ-JSF wins again.

Finally we remark that more precomputations in combination with the ϕ-JSF can be used to obtain further speedup. Avanzi [2] shows that allowing 10 precomputations one obtains an expansion needing only $3l/8$ additions and l doublings. His analysis applies also to the ϕ-JSF.

Acknowledgements

We are grateful to Takakazu Satoh for suggesting us to extend the GLV method to include ϕ-NAF expansions. We would also like to thank Roberto Maria Avanzi for useful comments on the preliminary version of this paper.

References

1. IEEE Std 1363-2000. *IEEE Standard Specifications for Public-Key Cryptography.* IEEE Computer Society, August 29, 2000.
2. R.M. Avanzi. On multi-exponentiation in cryptography. Technical Report 2002/154, Cryptology ePrint Archive, Available at: http://eprint.iacr.org/2002/154, 2002.
3. D.J. Bernstein. Pippenger's exponentiation algorithm. Available at: http://cr.yp.to/papers.html, 2002.
4. I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society.* Cambridge University Press, 2000.
5. H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics.* Springer, 1996.
6. H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve using Mixed Coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptography - Proceedings of ASIACRYPT 1998*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 1998.
7. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
8. Standard for Efficient Cryptography. Elliptic Curve Cryptography Ver.1.0. Technical report, Certicom, Available at: http://www.secg.org/drafts.html, 2001.
9. R. P. Gallant, J. L. Lambert, and S. A. Vanstone. Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.
10. D. M. Gordon. A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27(1):129–146, 1998.
11. N. Koblitz. CM-curves with good cryptographic properties. In Joan Feigenbaum, editor, *Advances in Cryptology - Proceedings of CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287, Berlin, 1991. Springer.
12. T. Lange. *Efficient Arithmetic on Hyperelliptic Koblitz Curves.* PhD thesis, University of Essen, 2001.
13. F. Morain and J. Olivos. Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains. *Inform. Theor. Appl.*, 24:531–543, 1990.
14. V. Müller. Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two. *Journal of Cryptology*, 11(4):219–234, 1998.
15. V. Müller. Efficient Point Multiplication for Elliptic Curves over Special Optimal Extension Fields. In Walter de Gruyter, editor, *Public-Key Cryptography and Computational Number Theory*, pages 197–207, Warschau, Poland, September 11-15, 2000 (2001).
16. National Institute of Standards and Technology. FIPS-186-2: Digital Signature Standard (DSS), January 2000. Available at http://csrc.nist.gov/publications/fips/.
17. Y-H. Park, S. Jeong, C. Kim, and J. Lim. An Alternate Decomposition of an Integer for Faster Point Multiplication on Certain Elliptic Curves. In D. Naccache and P. Paillier, editors, *Advances in Cryptology - Proceedings of PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.
18. G.W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.
19. F. Sica, M. Ciet, and J-J. Quisquater. Analysis of the Gallant-Lambert-Vanstone Method based on Efficient Endomorphisms: Elliptic and Hyperelliptic Curves. In

H. Heys and K. Nyberg, editors, *Proceedings of Selected Areas in Cryptography (SAC 2002)*, Lecture Notes in Computer Science. Springer, 2002. To appear.
20. N.P. Smart. Elliptic Curve Cryptosystems over Small Fields of Odd Characteristic. *Journal of Cryptology*, 12(2):141–151, 1999.
21. J. Solinas. Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography*, 19:195–249, 2000.
22. J. A. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - Proceedings of CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1997.
23. J.A. Solinas. Low-Weight Binary Representations for Pairs of Integers. Technical Report CORR 2001-41, CACR, Available at: www.cacr.math.uwaterloo.ca/ techreports/2001/corr2001-41.ps, 2001.
24. E.G. Straus. Addition chains of vectors (problem 5125). *American Mathematical Monthly 70*, pages 806–808, 1964.