Four-Dimensional Gallant-Lambert-Vanstone Scalar Multiplication

Patrick Longa[1] and Francesco Sica[2]

[1] Microsoft Research, USA
plonga@microsoft.com
[2] Nazarbayev University, Kazakhstan
francesco.sica@nu.edu.kz

Abstract. The GLV method of Gallant, Lambert and Vanstone (CRYPTO 2001) computes any multiple kP of a point P of prime order n lying on an elliptic curve with a low-degree endomorphism Φ (called GLV curve) over $\mathbb{F}_p$ as $kP = k_1P + k_2\Phi(P)$, with $\max\{|k_1|, |k_2|\} \leq C_1\sqrt{n}$, for some explicit constant $C_1 > 0$. Recently, Galbraith, Lin and Scott (EUROCRYPT 2009) extended this method to all curves over $\mathbb{F}_{p^2}$ which are twists of curves defined over $\mathbb{F}_p$. We show in this work how to merge the two approaches in order to get, for twists of any GLV curve over $\mathbb{F}_{p^2}$, a four-dimensional decomposition together with fast endomorphisms Φ, Ψ over $\mathbb{F}_{p^2}$ acting on the group generated by a point P of prime order n, resulting in a proven decomposition for any scalar $k \in [1, n]$ given by $kP = k_1P + k_2\Phi(P) + k_3\Psi(P) + k_4\Psi\Phi(P)$ with $\max_i(|k_i|) < C_2\, n^{1/4}$, for some explicit $C_2 > 0$. Remarkably, taking the best C_1, C_2, we obtain $C_2/C_1 < 412$, independently of the curve, ensuring in theory an almost constant relative speedup. In practice, our experiments reveal that the use of the merged GLV-GLS approach supports a scalar multiplication that runs up to 50% times faster than the original GLV method. We then improve this performance even further by exploiting the Twisted Edwards model and show that curves originally slower may become extremely efficient on this model. In addition, we analyze the performance of the method on a multicore setting and describe how to efficiently protect GLV-based scalar multiplication against several side-channel attacks. Our implementations improve the state-of-the-art performance of point multiplication for a variety of scenarios including side-channel protected and unprotected cases with sequential and multicore execution.

Keywords. Elliptic curves, GLV-GLS method, scalar multiplication, Twisted Edwards curve, side-channel protection, multicore computation.

1 Introduction

The Gallant-Lambert-Vanstone (GLV) method is a generic approach to speed up the computation of scalar multiplication on some elliptic curves defined over fields of large prime characteristic. Given a curve with a point P of prime order n, it consists essentially in an algorithm that finds a decomposition of an arbitrary scalar multiplication kP for $k \in [1, n]$ into two scalar multiplications, with the new scalars having only about half the bitlength of the original scalar. This immediately enables the elimination of half the doublings by employing the Straus-Shamir trick for simultaneous point multiplication.

Whereas the original GLV method as defined in [10] works on curves over $\mathbb{F}_p$ with an endomorphism of small degree (GLV curves), Galbraith-Lin-Scott (GLS) in [8] have shown that over $\mathbb{F}_{p^2}$ one can expect to find many more such curves by basically exploiting the action of the Frobenius endomorphism. One can therefore expect that on the particular GLV curves, this new insight will lead to improvements over $\mathbb{F}_{p^2}$. Indeed the GLS article itself considers four-dimensional decompositions on GLV curves with nontrivial automorphisms (corresponding to the degree one cases) but leaves the other cases open to investigation.

In this work, we generalize the GLS method to *all* GLV curves by exploiting fast endomorphisms Φ, Ψ over $\mathbb{F}_{p^2}$ acting on a cyclic group generated by a point P of prime order n to construct a proven decomposition with no heuristics involved for any scalar $k \in [1, n]$

$$kP = k_1P + k_2\Phi(P) + k_3\Psi(P) + k_4\Psi\Phi(P) \text{ with } \max_i(|k_i|) < Cn^{1/4}$$

for some explicitly computable C. In doing this we provide a reduction algorithm for the four-dimensional relevant lattice which runs in $O(\log^2 n)$ by implementing two Cornacchia-type algorithms [6, 22], one in $\mathbb{Z}$, the other in $\mathbb{Z}[i]$. The algorithm is remarkably simple to implement and allows us to demonstrate an improved $C = O(\sqrt{s})$ (compared to the value obtained with LLL which is only $\Omega(s^{3/2})$). Thus, it guarantees a relative speedup independent of the curve when moving from a two-dimensional to a four-dimensional GLV method over the same underlying field. If parallel computation is available then the computation of kP can possibly be implemented (close to) four times faster in this case. When moving from two-dimensional GLV over $\mathbb{F}_p$ to the four-dimensional case over $\mathbb{F}_{p^2}$, our method still guarantees a relative speedup that is *quasi*-uniform among all GLV curves (see Section 7 for details). In fact, we present experimental results on different GLV curves that demonstrate that the relative speedup between the original GLV method and the proposed method (termed GLV-GLS in the remainder) is as high as 50%.

Twisted Edwards curves [2] are efficient generalizations of Edwards curves [7], which exhibit high-performance arithmetic. By exploiting this curve model, Galbraith, Lin and Scott showed in [9] that the GLS method can be improved in practice a further 10%, approximately (see also [19, 18]). They also described how to write down j-invariant 0 and 1728 curves in Edwards form to combine a 4-dimensional decomposition with the fast arithmetic provided by this curve model. We exploit this approach and, most remarkably, lift the restriction to those special curves and show that in practice the GLV-GLS curves discussed in this work may achieve extremely high-performance and become virtually equivalent in terms of speed when written in Twisted Edwards form.

In the last years multiple works have incrementally shown the impact of using the GLS method for high performance [8, 19, 13]. However, it is still unclear how well the method behaves on settings where side-channel attacks are a threat. Since it is usually assumed that required countermeasures once in place degrade performance significantly, it is also unclear if the GLS method would retain its current superiority in the case of side-channel protected implementations. Here, we study this open problem and describe how to protect implementations based on the GLV-GLS method against timing attacks, cache attacks and similar ones and still achieve very high performance. The techniques discussed naturally apply to GLV-based implementations in general. Finally, we discuss different strategies to implement GLV-based scalar multiplication on modern multicore processors, and include the case in which countermeasures against side-channel attacks are required.

The presented implementations corresponding to the GLV-GLS method improve the state-of-the-art performance of point multiplication for all the cases under study: protected and unprotected versions with sequential and parallel execution. For instance, on one core of an Intel Core i7-2600 processor and at roughly 128 bits of security, we compute an *unprotected* scalar multiplication in only 91,000 cycles (which is 1.34 times faster than a previous result reported by Hu, Longa and Xu in [13]), and a *side-channel protected* scalar multiplication in only 137,000 cycles (which is 1.42 times faster than the protected implementation presented by Bernstein et al. in [3]).

Related Work: Recently, a paper by Zhou, Hu, Xu and Song [28] has shown that it is possible to combine the GLV and GLS approaches by introducing a three-dimensional version of the GLV method, which seems to work to a certain degree, with however no justification but through practical

implementations. The first author together with Hu and Xu [13] studied the case of curves with j-invariant 0 and provided a bound for this particular case. Our analysis supplements [13] by considering all GLV curves and providing a unified treatment.

2 The GLV Method

In this section we briefly summarize the GLV method following [25]. Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ and P be a point on this curve with prime order n such that the cofactor $h = \#E(\mathbb{F}_q)/n$ is small, say $h \leq 4$. Let us consider Φ a non trivial endomorphism defined over $\mathbb{F}_q$ and $X^2 + rX + s$ its characteristic polynomial. In all the examples r and s are actually small fixed integers and q is varying in some family. By hypothesis there is only one subgroup of order n in $E(\mathbb{F}_q)$, implying that $\Phi(P) = \lambda P$ for some $\lambda \in [0, n-1]$, since $\Phi(P)$ has order dividing the prime n. In particular, λ is obtained as a root of $X^2 + rX + s$ modulo n.

Define the group homomorphism (the GLV reduction map)

$$\begin{aligned} \mathfrak{f}\colon \mathbb{Z} \times \mathbb{Z} &\to \mathbb{Z}/n \\ (i, j) &\mapsto i + \lambda j \pmod{n} \ . \end{aligned}$$

Let $\mathcal{K} = \ker \mathfrak{f}$. It is a sublattice of $\mathbb{Z} \times \mathbb{Z}$ of rank 2 since the quotient is finite. Let $\Bbbk > 0$ be a constant (depending on the curve) such that we can find v_1, v_2 two linearly independent vectors of $\mathcal{K}$ satisfying $\max\{|v_1|, |v_2|\} < \Bbbk\sqrt{n}$, where $|\cdot|$ denotes the rectangle norm[3]. Express $(k, 0) = \beta_1 v_1 + \beta_2 v_2$, where $\beta_i \in \mathbb{Q}$. Then round β_i to the nearest integer $b_i = \lfloor \beta_i \rceil = \lfloor \beta_i + 1/2 \rfloor$ and let $v = b_1 v_1 + b_2 v_2$. Note that $v \in \mathcal{K}$ and that $u \stackrel{\text{def}}{=} (k, 0) - v$ is short. Indeed by the triangle inequality we have that

$$|u| \leq \frac{|v_1| + |v_2|}{2} < \Bbbk\sqrt{n} \ .$$

If we set $(k_1, k_2) = u$, then we get $k \equiv k_1 + k_2\lambda \pmod{n}$ or equivalently $kP = k_1 P + k_2 \Phi(P)$, with $\max(|k_1|, |k_2|) < \Bbbk\sqrt{n}$.

In [25], the optimal value of $\Bbbk$ (with respect to large values of n, i.e. large fields, keeping $X^2 + rX + s$ constant) is determined. Let $\Delta = r^2 - 4s$ be the discriminant of the characteristic polynomial of Φ. Then the optimal $\Bbbk$ is given by the following result[4] .

Theorem 1 ([25, Theorem 4]). *Assuming n is the norm of an element of $\mathbb{Z}[\Phi]$, then the optimal value of $\Bbbk$ is*

$$\Bbbk = \begin{cases} \dfrac{\sqrt{s}}{2}\Big(1 + \dfrac{1}{|\Delta|}\Big), & \text{if } r \text{ is odd,} \\ \dfrac{\sqrt{s}}{2}\sqrt{1 + \dfrac{4}{|\Delta|}}, & \text{if } r \text{ is even.} \end{cases}$$

3 The GLS Improvement

In 2009, Galbraith, Lin and Scott [8] realised that we do not need to have $\Phi^2 + r\Phi + s = 0$ in $\mathrm{End}(E)$ but only in a subgroup of $E(\mathbb{F})$ for a specific finite field $\mathbb{F}$. In particular, considering $\Psi = \mathrm{Frob}_p$

[3] The rectangle norm of (x, y) is by definition $\max(|x|, |y|)$. As remarked in [25], we can replace it by any other metric norm. We will use the term "short" to denote smallness in the rectangle norm.

[4] There is a mistake in [25] in the derivation of $\Bbbk$ for odd values of r. This affects [25, Corollary 1] for curves E_2 and E_3, where the correct values of $\Bbbk$ are respectively $2/3$ and $4\sqrt{2}/7$.

the p-Frobenius endomorphism of a curve E defined over $\mathbb{F}_p$, we know that $\Psi^m(P) = P$ for all $P \in E(\mathbb{F}_{p^m})$. While this tells nothing useful if $m = 1, 2$, it does offer new nontrivial relations for higher degree extensions. The case $m = 4$ is particularly useful here.

In this case if $P \in E(\mathbb{F}_{p^4}) \backslash E(\mathbb{F}_{p^2})$ then $\Psi^2(P) = -P$ and hence on the subgroup generated by P, Ψ satisfies the equation $X^2 + 1 = 0$. This implies that if $\Psi(P)$ is a multiple of P (which happens as soon as the order n of P is sufficiently large, say at least $2p$), we can apply the GLV approach and split again a scalar multiplication as $kP = k_1 P + k_2 \Psi(P)$, with $\max(|k_1|, |k_2|) = O(\sqrt{n})$. Contrast this with the characteristic polynomial of Ψ which is $X^2 - a_p X + p$ for some integer a_p, a non-constant polynomial to which we cannot apply as efficiently the GLV paradigm.

For efficiency reasons however one does not work with $E/\mathbb{F}_{p^4}$ directly but with $E'/\mathbb{F}_{p^2}$ isomorphic to E *over* $\mathbb{F}_{p^4}$ but not over $\mathbb{F}_{p^2}$, that is, a quadratic twist over $\mathbb{F}_{p^2}$. In this case, it is possible that $\#E'(\mathbb{F}_{p^2}) = n \geq (p-1)^2$ be prime. Furthermore, if $\psi \colon E' \to E$ is an isomorphism defined over $\mathbb{F}_{p^4}$, then the endomorphism $\Psi = \psi \operatorname{Frob}_p \psi^{-1} \in \operatorname{End}(E')$ satisfies the equation $X^2 + 1 = 0$ and if $p \equiv 5 \pmod 8$ it can be defined over $\mathbb{F}_p$.

This idea is at the heart of the GLS approach, but it only works for curves over $\mathbb{F}_{p^m}$ with $m > 1$, therefore it does not generalise the original GLV method but rather complements it.

4 Combining GLV and GLS

Let $E/\mathbb{F}_p$ be a GLV curve. As in Section 3, we will denote by $E'/\mathbb{F}_{p^2}$ a quadratic twist $\mathbb{F}_{p^4}$-isomorphic to E via the isomorphism $\psi \colon E \to E'$. We also suppose that $\#E'(\mathbb{F}_{p^2}) = nh$ where n is prime and $h \leq 4$. We then have the two endomorphisms of E', $\Psi = \psi \operatorname{Frob}_p \psi^{-1}$ and $\Phi = \psi \phi \psi^{-1}$, with ϕ the GLV endomorphism coming with the definition of a GLV curve. They are both defined over $\mathbb{F}_{p^2}$, since if σ is the nontrivial Galois automorphism of $\mathbb{F}_{p^4}/\mathbb{F}_{p^2}$, then $\psi^\sigma = -\psi$, so that $\Psi^\sigma = \psi^\sigma \operatorname{Frob}_p^\sigma \big(\psi^{-1}\big)^\sigma = (-\psi) \operatorname{Frob}_p(-\psi^{-1}) = \Psi$, meaning that $\Psi \in \operatorname{End}_{\mathbb{F}_{p^2}}(E')$. Similarly for Φ, where we are using the fact that $\phi \in \operatorname{End}_{\mathbb{F}_p}(E)$. Notice that $\Psi^2 + 1 = 0$ and that Φ has the same characteristic polynomial as ϕ. Furthermore, since we have a large subgroup $\langle P \rangle \subset E'(\mathbb{F}_{p^2})$ of prime order, $\Phi(P) = \lambda P$ and $\Psi(P) = \mu P$ for some $\lambda, \mu \in [1, n-1]$. We will assume that Φ and Ψ, when viewed as algebraic integers, generate disjoint quadratic extensions of $\mathbb{Q}$. In particular, we are not dealing with Example 1 from Appendix A, but this can be treated separately with a quartic twist as described in Appendix B of the full version of this article [21].

Consider the biquadratic (Galois of degree 4, with Galois group $\mathbb{Z}/2 \times \mathbb{Z}/2$) number field $K = \mathbb{Q}(\Phi, \Psi)$. Let $\mathfrak{o}_K$ be its ring of integers. The following analysis is inspired by [25, Section 8].

We have $\mathbb{Z}[\Phi, \Psi] \subseteq \mathfrak{o}_K$. Since the degrees of Φ and Ψ are much smaller when compared to n, the prime n is unramified in K and the existence of λ and μ above means that n splits in $\mathbb{Q}(\Phi)$ and $\mathbb{Q}(\Psi)$, namely that n splits completely in K. There exists therefore a prime ideal $\mathfrak{n}$ of $\mathfrak{o}_K$ dividing $n\mathfrak{o}_K$, such that its norm is n. We can also suppose that $\Phi \equiv \lambda \pmod{\mathfrak{n}}$ and $\Psi \equiv \mu \pmod{\mathfrak{n}}$. The four-dimensional GLV-GLS method works as follows.

Consider the GLV-GLS reduction map F defined by

$$\begin{aligned} F \colon \mathbb{Z}^4 &\to \mathbb{Z}/n \\ (x_1, x_2, x_3, x_4) &\mapsto x_1 + x_2\lambda + x_3\mu + x_4\lambda\mu \pmod n \ . \end{aligned}$$

If we can find four linearly independent vectors $v_1, \ldots, v_4 \in \ker F$, with $\max_i |v_i| \leq C n^{1/4}$ for some constant $C > 0$, then for any $k \in [1, n-1]$ we write

$$(k, 0, 0, 0) = \sum_{j=1}^{4} \beta_j v_j \ ,$$

with $\beta_j \in \mathbb{Q}$. As in the GLV method one sets $v = \sum_{j=1}^{4} \lfloor \beta_j \rceil v_j$ and

$$u = (k,0,0,0) - v = (k_1, k_2, k_3, k_4) \ .$$

We then get

$$kP = k_1 P + k_2 \Phi(P) + k_3 \Psi(P) + k_4 \Psi\Phi(P) \quad \text{with } \max_i(|k_i|) \leq 2Cn^{1/4} \ . \tag{1}$$

We focus next on the study of $\ker F$ in order to find a reduced basis v_1, v_2, v_3, v_4 with an explicit C. We can factor the GLV-GLS map F as

$$\begin{array}{ccccc} \mathbb{Z}^4 & \xrightarrow{f} & \mathbb{Z}[\Phi,\Psi] & \xrightarrow[\bmod \mathfrak{n}\cap\mathbb{Z}[\Phi,\Psi]]{\text{reduction}} & \mathbb{Z}/n \\ (x_1,x_2,x_3,x_4) & \longmapsto & x_1 + x_2\Phi + x_3\Psi + x_4\Phi\Psi & \longmapsto & x_1 + x_2\lambda + x_3\mu + x_4\lambda\mu \pmod n \ . \end{array}$$

Notice that the kernel of the second map (reduction mod $\mathfrak{n} \cap \mathbb{Z}[\Phi,\Psi]$) is exactly $\mathfrak{n} \cap \mathbb{Z}[\Phi,\Psi]$. This can be seen as follows. The reduction map factors as

$$\mathbb{Z}[\Phi,\Psi] \longrightarrow \mathfrak{o}_K \longrightarrow \mathfrak{o}_K/\mathfrak{n} \cong \mathbb{Z}/n$$

where the first arrow is inclusion, the second is reduction mod $\mathfrak{n}$, corresponding to reducing the x_i's mod $\mathfrak{n} \cap \mathbb{Z} = n\mathbb{Z}$ and using $\Phi \equiv \lambda, \Psi \equiv \mu \pmod{\mathfrak{n}}$. But the kernel of this map consists precisely of elements of $\mathbb{Z}[\Phi,\Psi]$ which are in $\mathfrak{n}$, and that is what we want.

Moreover, since the reduction map is surjective, we obtain an isomorphism $\mathbb{Z}[\Phi,\Psi]/\mathfrak{n} \cap \mathbb{Z}[\Phi,\Psi] \cong \mathbb{Z}/n$ which says that the index of $\mathfrak{n} \cap \mathbb{Z}[\Phi,\Psi]$ inside $\mathbb{Z}[\Phi,\Psi]$ is n. Since the first map f is an isomorphism, we get that $\ker F = f^{-1}(\mathfrak{n} \cap \mathbb{Z}[\Phi,\Psi])$ and that $\ker F$ has index $[\mathbb{Z}^4 \colon \ker F] = n$ inside $\mathbb{Z}^4$.

We can also produce a basis of $\ker F$ by the following observation. Let $\Phi' = \Phi - \lambda$, $\Psi' = \Psi - \mu$, hence $\Phi'\Psi' = \Phi\Psi - \lambda\Psi - \mu\Phi + \lambda\mu$. In matrix form,

$$\begin{pmatrix} 1 \\ \Phi' \\ \Psi' \\ \Phi'\Psi' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\lambda & 1 & 0 & 0 \\ -\mu & 0 & 1 & 0 \\ \lambda\mu & -\mu & -\lambda & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \Phi \\ \Psi \\ \Phi\Psi \end{pmatrix}$$

Since the determinant of the square matrix is 1, we deduce that $\mathbb{Z}[\Phi,\Psi] = \mathbb{Z}[\Phi',\Psi']$. But in this new basis, we claim that

$$\mathfrak{n} \cap \mathbb{Z}[\Phi',\Psi'] = n\mathbb{Z} + \mathbb{Z}\Phi' + \mathbb{Z}\Psi' + \mathbb{Z}\Phi'\Psi' \ .$$

Indeed, reverse inclusion ($\supseteq$) is easy since $\Phi', \Psi', \Phi'\Psi' \in \mathfrak{n}$ and so is n, because $\mathfrak{n}$ divides $n\mathfrak{o}_K$ is equivalent to $\mathfrak{n} \supseteq n\mathfrak{o}_K$. On the other hand, the index of both sides in $\mathbb{Z}[\Phi',\Psi']$ is n, which can only happen, once an inclusion is proved, if the two sides are equal. Using the isomorphism f, we see that a basis of $\ker F \subset \mathbb{Z}^4$ is therefore given by

$$w_1 = (n,0,0,0), w_2 = (-\lambda,1,0,0), w_3 = (-\mu,0,1,0), w_4 = (\lambda\mu,-\mu,-\lambda,1) \ .$$

The LLL algorithm [17] then finds, for a given basis $w_1, \ldots, w_4$ of $\ker F$, a reduced[5] basis $v_1, \ldots, v_4$ in polynomial time (in the logarithm of the norm of the w_i's) such that (cf. [5, Theorem 2.6.2 p.85])

$$\prod_{i=1}^{4} |v_i| \leq 8\,[\mathbb{Z}^4 \colon \ker F] = 8n \ . \tag{2}$$

[5] The estimates are usually given for the Euclidean norm of the vectors. But it is easy to see that the rectangle norm is upper bounded by the Euclidean norm.

Lemma 1. *Let Φ and Ψ be as defined at the beginning of this section,*

$$\begin{aligned} \mathcal{N}\colon \mathbb{Z}^4 &\to \mathbb{Z} \\ (x_1,x_2,x_3,x_4) &\mapsto \sum_{\substack{i_1,i_2,i_3,i_4\geq 0 \\ i_1+i_2+i_3+i_4=4}} b_{i_1,i_2,i_3,i_4} x_1^{i_1}x_2^{i_2}x_3^{i_3}x_4^{i_4} \end{aligned}$$

be the norm of an element $x_1 + x_2\Phi + x_3\Psi + x_4\Phi\Psi \in \mathbb{Z}[\Phi,\Psi]$, where the b_{i_1,i_2,i_3,i_4}'s lie in $\mathbb{Z}$. Then, for any nonzero $v \in \ker F$, one has

$$|v| \geq \frac{n^{1/4}}{\Big(\sum\limits_{\substack{i_1,i_2,i_3,i_4 \\ i_1+i_2+i_3+i_4=4}} |b_{i_1,i_2,i_3,i_4}|\Big)^{1/4}} \ . \tag{3}$$

Proof. For $v \in \ker F$ we have $\mathcal{N}(v) \equiv 0 \pmod{n}$ and if $v \neq 0$ we must therefore have $|\mathcal{N}(v)| \geq n$. On the other hand, if we did not have (3), then every component of v would be strictly less than the right-hand side and plugging this upper bound in the definition of $|\mathcal{N}(v)|$ would yield a quantity $< n$, a contradiction. □

Let B be the denominator of the right-hand side of (3), then (2) and (3) imply that

$$|v_i| \leq 8B^3\, n^{1/4} \quad i = 1,2,3,4 \ . \tag{4}$$

Remark 1. In our case, where $\Psi^2 + 1 = 0$ and $\Phi^2 + r\Phi + s = 0$, we get as norm function

$$\begin{aligned} x_1^4 + s^2x_2^4 + x_3^4 + s^2x_4^4 - 2rx_1^3x_2 - 2rsx_1x_2^3 - 2rx_3^3x_4 - 2rsx_3x_4^3 + \\ (r^2+2s)x_1^2x_2^2 + 2x_1^2x_3^2 + (r^2-2s)x_1^2x_4^2 + (r^2-2s)x_2^2x_3^2 + 2s^2x_2^2x_4^2 + (r^2+2s)x_3^2x_4^2 \\ - 2rx_1^2x_3x_4 - 2rsx_2^2x_3x_4 - 2rx_1x_2x_3^2 - 2rsx_1x_2x_4^2 + 8sx_1x_2x_3x_4 \ , \end{aligned}$$

and therefore

$$B = \big(4 + 4s^2 + 8s + 8|r| + 8|r|s + 2(r^2+2s) + 2|r^2-2s|\big)^{1/4} \ . \tag{5}$$

From (1) and (4) we have proved the following theorem.

Theorem 2. *Let $E/\mathbb{F}_p$ be a GLV curve and $E'/\mathbb{F}_{p^2}$ a twist, together with the two efficient endomorphisms Φ and Ψ, where everything is defined as at the start of this section. Suppose that the minimal polynomial of Φ is $X^2 + rX + s = 0$. Let $P \in E'(\mathbb{F}_{p^2})$ be a generator of the large subgroup of prime order n. There exists an efficient algorithm, which for any $k \in [1,n]$ finds integers k_1,k_2,k_3,k_4 such that*

$$kP = k_1P + k_2\Phi(P) + k_3\Psi(P) + k_4\Psi\Phi(P) \quad \textit{with } \max_i(|k_i|) \leq 16B^3n^{1/4}$$

and

$$B = \big(4 + 4s^2 + 8s + 8|r| + 8|r|s + 2(r^2+2s) + 2|r^2-2s|\big)^{1/4} \ .$$

4.1 Uniform Improvements

The previous analysis is only the first step of our work. It shows that the GLV-GLS method works as predicted in a four-way decomposition on twists of GLV curves over $\mathbb{F}_{p^2}$. However, the constant B^3 involved is rather large and, hence, does not guarantee a non-negligible gain when switching from 2 to 4 dimensions (especially on those GLV curves with more complicated endomorphism rings). A much deeper argument, which we develop in the full version of this article, allows us to prove the following result.

Theorem 3. *When performing an optimal lattice reduction on* $\ker F$*, it is possible to decompose any* $k \in [1, n]$ *into integers* k_1, k_2, k_3, k_4 *such that*

$$kP = k_1 P + k_2 \Phi(P) + k_3 \Psi(P) + k_4 \Psi\Phi(P) \ ,$$

with $\max_i(|k_i|) < 103(\sqrt{1 + |r| + s})\, n^{1/4}$.

The significance of this theorem lies in the *uniform* improvement of the constant $16B^3$, which is $\Omega(s^{3/2})$ in Theorem 2, to a value that is an absolute constant times greater than the minimal bound for the 2-dimensional GLV method (Theorem 1). Hence, this guarantees in practice a quasi-uniform improvement when switching from 2-dimensional to 4-dimensional GLV independently of the curve.

To prove Theorem 3, first note that Lemma 1 gives a rather poor bound when applied to more than one vector, as is done three times for the proof of Theorem 2. A more direct treatment of the reduced vectors of $\ker F$ becomes necessary, and this is done via a modification of the original GLV approach. This results into a new, easy-to-implement lattice reduction algorithm which employs two Cornacchia-type algorithms [5, Section 1.5.2], one in $\mathbb{Z}$ (as in the original GLV method), the other one in $\mathbb{Z}[i]$ (Gaussian Cornacchia). The new algorithm is presented in Appendix B. The main difficulty lies in controlling arguments of complex numbers in the Gaussian Cornacchia algorithm and is technically rather delicate. This difficulty does not exist in the original GLV algorithm, as taking absolute values suffices to get the desired bounds. We refer to the full version [21] for details.

Remark 2. In the case of the LLL algorithm, we have not managed to demonstrate a bound as good as the one obtained with our lattice reduction algorithm.

Remark 3. Nguyen and Stehlé [23] have produced an efficient lattice reduction in four dimensions which finds successive minima and hence produces a decomposition with relatively good bounds. Our algorithm represents a very simple and easy-to-implement alternative that may be ideal for certain cryptographic libraries.

5 GLV-GLS using the Twisted Edwards Model

The GLV-GLS method can be sped up in practice by writing down GLV-GLS curves in the Twisted Edwards model. Note that arithmetic on j-invariant 0 Weierstrass curves is already very efficient. However, some GLV curves do not exhibit such high-speed arithmetic. In particular, curves in Examples 3-6 from Appendix A have Weierstrass coefficients $a_4 \cdot a_6 \neq 0$ for curve parameters a_4 and a_6 and hence they have more expensive point doubling (even more if we consider the extra multiplication by the twisted parameter u when using the GLS method). So the impact of using Twisted Edwards is expected to be especially significant for these curves. In fact, if we consider that suitable parameters can be always chosen the use of Twisted Edwards curves isomorphic to the original Weierstrass GLV-GLS curves *uniformizes* the performance of all of them.

Let us illustrate how to produce a Twisted Edwards GLV-GLS curve with the GLV curve from Example 4, Appendix A. First, consider its quadratic twist over $\mathbb{F}_{p^2}$

$$E'/\mathbb{F}_{p^2} : \ x^3 - \frac{15}{2}u^2 x - 7u^3 = (x + 2u) \cdot (x^2 - 2ux - \frac{7}{2}u^2)$$

The change of variables $x_1 = x + 2u$ transforms E' into

$$y^2 = x_1^3 - 6ux_1^2 + \frac{9u^2}{2}x_1 \ .$$

Let $\beta = 3u/\sqrt{2} \in \mathbb{F}_{p^2}$ and substitute $x_1 = \beta x'$ to get

$$\frac{1}{\beta^3} y^2 = x'^3 - \frac{6u}{\beta} x'^2 + x'$$

and this is a Montgomery curve $M_{A,B} : Bv^2 = u^3 + Au^2 + u$, where $A \neq \pm 2, B \neq 0$, with

$$B = \frac{1}{\beta^3} = \frac{2\sqrt{2}}{27u^3} \ , \quad A = -\frac{6u}{\beta} = -2\sqrt{2} \ .$$

The corresponding Twisted Edwards GLV-GLS curve is then $E_{a,d}\colon ax^2 + y^2 = 1 + dx^2y^2$ with

$$a = \frac{A+2}{B} = 27u^3 \left(\frac{\sqrt{2}}{2} - 1 \right) \ , \quad d = \frac{A-2}{B} = -27u^3 \left(\frac{\sqrt{2}}{2} + 1 \right) \ .$$

The map $E' \to E_{a,d}$ is

$$(x, y) \mapsto \left(\frac{x + 2u}{\beta y}, \frac{x + 2u - \beta}{x + 2u + \beta} \right) = (X, Y)$$

with inverse

$$(X, Y) \mapsto \left(\frac{\beta - 2u + (\beta + 2u)Y}{1 - Y}, \frac{1 + Y}{(1 - Y)X} \right) \ .$$

We now specify the formulas for Φ and Ψ, obtained by composing these endomorphisms on the Weierstrass model with the birational maps above. We found an extremely appealing expression in the case when $u = 1 + i$ and $i^2 = -1$. Then $\beta = 3u/\sqrt{2} = 3\zeta_8$ where ζ_8 is a primitive 8th root of unity. We have

$$\Phi(X, Y) = \left(-\frac{(\zeta_8^3 + 2\zeta_8^2 + \zeta_8)XY^2 + (\zeta_8^3 - 2\zeta_8^2 + \zeta_8)X}{2Y}, \frac{(\zeta_8^2 - 1)Y^2 + 2\zeta_8^3 - \zeta_8^2 + 1}{(2\zeta_8^3 + \zeta_8^2 - 1)Y^2 - \zeta_8^2 + 1} \right)$$

and

$$\Psi(X, Y) = \left(\zeta_8 X^p, \frac{1}{Y^p} \right) \ .$$

In this case

$$a = 54(\zeta_8^3 - \zeta_8^2 + 1) \ , \quad d = -54(\zeta_8^3 + \zeta_8^2 - 1) \ .$$

Finally, one would want to use the efficient formulas given in [12] for the case $a = -1$. After ensuring that $-a$ be a square in $\mathbb{F}_{p^2}$, we use the map $(x, y) \mapsto (x/\sqrt{-a}, y)$ to convert to the isomorphic curve $-x^2 + y^2 = 1 + d'x^2y^2$, where $d' = -d/a$.

6 Side-Channel Protection and Parallelization of the GLV-GLS Method

Given the potential threat posed by attacks that exploit timing information to deduce secret keys ([16, 4]), many works have proposed countermeasures to minimize the risks and achieve the so-called constant-time execution during cryptographic computations. In general, to avoid leakage the execution flow should be independent of the secret key. This means that conditional branches and secret-dependent table lookup indices should be avoided [15]. There are *five* key points that are especially vulnerable during the computation of scalar multiplication: inversion, modular reduction in field operations, precomputation, scalar recoding and double-and-add execution.

A well-known technique that is secure and easy to implement for inverting any field element a consists of computing the exponentiation $a^{p-2} \mod p$ using a short addition chain for $p-2$.

To protect field operations, one may exploit conditional move instructions typically found on modern x86 and x64 processors (a.k.a. `cmove`). Since conditional checks happen during operations such as addition and subtraction as part of the reduction step it is standard practice to replace conditional branches with the conditional move instruction. Luckily, these conditional branches are highly unpredictable and, hence, the substitution above does not only makes the execution constant-time but also more efficient in most cases. An exception happens when performing modular reduction during a field multiplication or squaring, where a final correction step could happen very rarely and hence a conditional branch may be more efficient.

In the case of precomputation, recent work by [15] and later by [3] showed how to enable the use of precomputed points by employing constant-time table lookups that mask the extraction of points. In our implementations (see Section 7), we exploit a similar approach based on `cmove` and conditional vector instructions instead, which is expected to achieve higher performance on some platforms than implementations based on logical instructions (see Listing 1 in [15]). Note that it is straightforward to enable the use of signed-digit representations that allow negative points by performing a second table lookup between the point selected in the first table lookup and its negated value.

To protect the scalar recoding and its corresponding double-and-add algorithm, one needs a regular pattern execution. Based on a method by [24], Joye and Tunstall [14] proposed a constant-time recoding that supports a regular execution double-and-add algorithm that exploits precomputations. The nonzero density of the method is $1/(w-1)$, where w is the window width. Therefore, there is certain loss in performance in comparison with an unprotected version with nonzero density $1/(w+1)$. In GLV-based implementations one has to deal with more than one scalar, and these scalars are scanned simultaneously during multi-exponentiation. So there are two issues that arise. First, how are the several scalars aligned with respect to their zero and nonzero digit representation?, and second, how do we guarantee the same representation length for all scalars so that no dummy operations are required? The first issue is inherently solved by the recoding algorithm itself. The input is always an odd number, which means that, from left to right, one obtains the execution pattern $(w-1)$ doublings, d additions, $(w-1)$ doublings, d additions, $\ldots$, $(w-1)$ doublings and d additions, for d-dimensional GLV. For dealing with even numbers, one may employ the technique described in [14] in a constant-time fashion, namely, scalars k_i that are even are replaced by k_i+1 and scalars that are odd are replaced by k_i+2 (the correction, also constant-time, is performed after the scalar multiplication computation using d point additions). Solution to the second issue was also hinted by [14]. The reader is referred to the full paper version for the modified recoding algorithm that outputs a regular pattern representation with fixed length. Note that in the case of Twisted Edwards one can alternatively use *unified* addition formulas that also work for doubling (see [2, 12] for details). However, our analysis indicates that this approach is consistently slower because of the high cost of these unified formulas in comparison to doubling and the extra cost incurred by the increase in constant-time table lookup accesses.

6.1 Multicore Computation and its Side-Channel Protection

Parallelization of scalar multiplication over prime fields is particularly difficult on modern multicore processors. This is due to the difficulty to perform point operations concurrently when executing

the double-and-add algorithm from left to right. From right to left parallelization is easier but performance is hurt because the use of precomputations is cumbersome. Hence, parallelization should be ideally performed at the field arithmetic level. Unfortunately, current multicore processors still impose a severe overhead for thread creation/destruction. During our tests we observed overheads of a few thousands of cycles on modern 64-bit CPUs (that is, much more costly than a point addition or doubling). Given this limitation, for the GLV method it seems the ideal approach (from a speed perspective) to let each core manage a separate scalar multiplication with k_i. This is simple to implement, minimizes thread management overhead and also eases the task of protecting the implementation against side-channel attacks since each scalar can be recoded using Algorithm 4 [21, App. E]. Using d cores, the total cost of a protected d-dimensional GLV l-bit scalar multiplication (disregarding precomputation) is about l/d doublings and $l/((w-1)\cdot d)$ mixed additions. A somewhat slower approach (but more power efficient) would be to let one core manage all doublings and let one or two extra cores manage the additions corresponding to nonzero digits. For instance, for dimension four and three cores the total cost (disregarding precomputation) is about l/d doublings and $l/((w-1)\cdot d)$ *general* additions, always that the latency of $(w-1)$ doublings be equivalent or greater than the addition part (otherwise, the cost is dominated by non-mixed additions).

7 Performance Analysis and Experimental Results

For our analysis and experiments, we consider the *five* curves below: two GLV curves in Weierstrass form with and without nontrivial automorphisms, their corresponding GLV-GLS counterparts and one curve in Twisted Edwards form isomorphic to the GLV-GLS curve E'_3 (see below).

- GLV-GLS curve with j-invariant 0 in Weierstrass form $E'_1/\mathbb{F}_{p_1^2} : y^2 = x^3 + 9u$, where $p_1 = 2^{127} - 58309$ and $\#E'_1(\mathbb{F}_{p_1^2}) = r$, where r is a 254-bit prime. We use $\mathbb{F}_{p_1^2} = \mathbb{F}_{p_1}[i]/(i^2+1)$ and $u = 1 + i \in \mathbb{F}_{p_1^2}$. E'_1 is the quadratic twist of the curve in Example 2, Appendix A. $\Phi(x,y) = \lambda P = (\xi x, y)$ and $\Psi(x,y) = \mu P = (u^{(1-p)/3}x^p, u^{(1-p)/2}y^p)$, where $\xi^3 = 1 \mod p_1$. We have that $\Phi^2 + \Phi + 1 = 0$ and $\Psi^2 + 1 = 0$.
- GLV curve with j-invariant 0 in Weierstrass form $E_2/\mathbb{F}_{p_2} : y^2 = x^3 + 2$, where $p_2 = 2^{256} - 11733$ and $\#E_2(\mathbb{F}_{p_2})$ is a 256-bit prime. This curve corresponds to Example 2, Appendix A.
- GLV-GLS curve in Weierstrass form $E'_3/\mathbb{F}_{p_3^2} : y^2 = x^3 - 15/2\ u^2x - 7u^3$, where $p_3 = 2^{127} - 5997$ and $\#E'_3(\mathbb{F}_{p_3^2}) = 8r$, where r is a 251-bit prime. We use $\mathbb{F}_{p_3^2} = \mathbb{F}_{p_3}[i]/(i^2+1)$ and $u = 1 + i \in \mathbb{F}_{p_3^2}$. E'_3 is the quadratic twist of a curve isomorphic to the one in Example 4, Appendix A. The formula for $\Phi(x,y) = \lambda P$ can be easily derived from $\psi(x,y)$, and $\Psi(x,y) = \mu P = (u^{(1-p)}x^p, u^{3(1-p)/2}y^p)$. It can be verified that $\Phi^2 + 2 = 0$ and $\Psi^2 + 1 = 0$.
- GLV-GLS curve in Twisted Edwards form $E'_{T3}/\mathbb{F}_{p_3^2} : -x^2 + y^2 = 1 + dx^2y^2$, where $d = 170141183460469231731687303715884099728 + 116829086847165810221872975542241037773i$, $p_3 = 2^{127} - 5997$ and $\#E'_{T3}(\mathbb{F}_{p_3^2}) = 8r$, where r is a 251-bit prime. We use again $\mathbb{F}_{p_3^2} = \mathbb{F}_{p_3}[i]/(i^2+1)$ and $u = 1 + i \in \mathbb{F}_{p_3^2}$. E'_{T3} is isomorphic to curve E'_3 above and was obtained following the procedure in Section 5. The formulas for $\Phi(x,y)$ and $\Psi(x,y)$ are also given in Section 5. It can be verified that $\Phi^2 + 2 = 0$ and $\Psi^2 + 1 = 0$.
- GLV curve $E_4/\mathbb{F}_{p_4} : y^2 = x^3 - 15/2\ x - 7$, where $p_4 = 2^{256} - 45717$ and $\#E_4(\mathbb{F}_{p_4}) = 2r$, where r is a 256-bit prime. This curve is isomorphic to the curve in Example 4, Appendix A.

Let us first analyze the performance of the GLV-GLS method over $\mathbb{F}_{p^2}$ in comparison with the traditional 2-GLV case over $\mathbb{F}_p$. We assume the use of a pseudo-Mersenne prime with form $p = 2^m - c$,

for small c (for our targeted curves, groups with (near) prime order cannot be constructed using the attractive Mersenne prime $p = 2^{127} - 1$). Given that we have a proven ratio $C_2/C_1 < 412$ that is independent of the curve, the only values left that could affect significantly a uniform speedup between GLV-GLS and 2-GLV are the quadratic non-residue β used to build $\mathbb{F}_{p^2}$ as $\mathbb{F}_p[i]/(i^2 - \beta)$, the value of the twisted parameter u and the cost of applying the endomorphisms Φ and Ψ. In particular, if $|\beta| > 1$ a few extra additions (or a multiplication by a small constant) are required per $\mathbb{F}_{p^2}$ multiplication and squaring. Luckily, for all the GLV curves listed in Appendix A one can always use a suitably chosen modulus p so that $|\beta|$ can be one or at least very close to it. Similar comments apply to the twisted parameter u. In this case, the extra cost (equivalent to a few additions) is added to the cost of point doubling always that the curve parameter a in the Weierstrass equation be different to zero (e.g., it does not affect j-invariant 0 curves). In the case of Twisted Edwards, we applied a better strategy, that is, we eliminated the twisted parameter u in the isomorphic curve. The cost of applying Φ and Ψ does depend on the chosen curve and it could be relatively expensive. If computing $\Phi(P)$, $\Psi(P)$ or $\Psi\Phi(P)$ is more expensive than point addition then its use can be limited to only one application (i.e., multiples of those values −if using precomputations− should be computed with point additions). Further, the extra cost can be minimized by choosing the optimal window width for each k_i.

To illustrate how the parameters above affect the performance gain we detail in Table 1 estimates for the cost of computing scalar multiplication with our representative curves. In the remainder, we use the following notation: M, S, A and I represent field multiplication, squaring, addition and inversion over $\mathbb{F}_p$, respect., and m, s, a and i represent the same operations over $\mathbb{F}_{p^2}$. Side-channel protected multiplication and squaring are denoted by m_s and s_s. We consider the cost of addition, substraction, negation, multiplication by 2 and division by 2 as equivalent. For the targeted curves in Weierstrass form, a mixed addition consists of 8 multiplications, 3 squarings and 7 additions, and a general addition consists of 12 multiplications, 4 squarings and 7 additions. For E'_1 and E_2, a doubling consists of 3 multiplications, 4 squarings and 7 additions, and for E'_3 and E_4, a doubling consists of 3 multiplications, 6 squarings and 12 additions. For Twisted Edwards we consider the use of mixed homogeneous/extended homogeneous projective coordinates [12]. In this case, a mixed addition consists of 7 multiplications and 7 additions, a general addition consists of 8 multiplications and 6/7 additions and a doubling consists of 4 multiplications, 3 squarings and 5 additions. We assume the use of interleaving [10] with width-w non-adjacent form (wNAF) and the use of the LM scheme for precomputing points on the Weierstrass curves [20] (see also [18, Ch. 3]).

Table 1. Operation counts and performance for scalar multiplication at approximately 128 bits of security. To determine the total costs we consider 1i=66m, 1s=0.76m and 1a=0.18m for E'_1, E'_3 and E'_{T3}; and 1I=290M, 1S=0.85M and 1A=0.18M for E_2 and E_4. The cost ratio of multiplications over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ is M/m=0.91. These values and the performance figures (in cycles) were obtained by benchmarking full implementations on a single core of a 3.4GHz Intel Core i7-2600 (Sandy Bridge) processor.

Curve	Method	Operation Count	Total Cost	Gain	Performance	Gain
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 32pts.	2i + 617m + 404s + 847a	1209m	51%	99,000cc	53%
$E_2(\mathbb{F}_{p_2})$	2-GLV, 16pts.	1I + 904M + 690S + 1240A	2004M≈1824m	-	151,000cc	-
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	1i + 742m + 225s + 767a	1117m	97%	91,000cc	102%
$E'_3(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	2i + 678m + 581s + 1200a	1468m	50%	121,000cc	52%
$E_4(\mathbb{F}_{p_4})$	2-GLV, 16pts.	1I + 950M + 970S + 1953A	2416M≈2199m	-	184,000cc	-

According to our theoretical estimates, it is expected that the relative speedup when moving from 2-GLV to GLV-GLS be as high as 50%, approximately. To confirm our findings, we realized

full implementations of the methods. Experimental results, also displayed in Table 1, closely follow our estimates and confirm that speedups in practice are about 52%. Most remarkably, the use of the Twisted Edwards model pushes performance even further. In Table 1, the expected gains for E'_{T3} are 31% and 97% in comparison with 4-GLV-GLS and 2-GLV in Weierstrass form (respect.). In practice, we achieved similar speedups, namely, 33% and 102% (respect.). Likewise, a rough analysis indicates that a Twisted Edwards GLV-GLS curve for a j-invariant 0 curve would achieve roughly similar speed to E'_{T3}, which means that in comparison to its corresponding Weierstrass counterpart the gains are 9% and 66% (respect.). This highlights the impact of using Twisted Edwards especially over those GLV-GLS curves relatively slower in the Weierstrass model. Timings were registered on a single core of a 3.4GHz Intel Core i7-2600 (Sandy Bridge) processor.

Let us now focus on curves E'_1, E_2 and E'_{T3} to assess performance of implementations targeting *four* scenarios of interest: unprotected and side-channel protected versions with sequential and multicore execution. Operation counts for computing a scalar multiplication at approximately 128 bits of security for the different cases are displayed in Table 2. The techniques to protect and parallelize our implementations are described in Section 6. In particular, the execution flow and memory address access of side-channel protected versions are not secret and are fully independent of the scalar. For our versions running on several cores we used OpenMP. We use an implementation in which each core is in charge of one scalar multiplication with k_i. Given the high cost of thread creation/destruction this approach guarantees the fastest computation in our case (see Section 6 for a discussion). Note that these multicore figures are only relevant for scenarios in which latency rather than throughput is targeted. Finally, we consider the cost of constant-time table lookups (denoted by t) given its non-negligible cost in protected implementations.

Table 2. Operation counts for scalar multiplication at approximately 128 bits of security using curves E'_1, E_2 and E'_{T3} in up to four variants: unprotected and side-channel protected implementations with sequential and multicore execution. To determine the total costs we consider 1i=66m, 1s=0.76m and 1a=0.18m for unprotected versions of E'_1 and E'_{T3}; 1i=79m$_s$, 1s$_s$=0.81m$_s$ and 1a=0.17m$_s$ for protected versions of E'_1 and E'_{T3}; t=0.83m$_s$ for E'_1 (32pts.); t=1.28m$_s$ for E'_{T3} (36pts.); t=0.78m$_s$ for E'_{T3} (20pts.); and 1I=290M, 1S=0.85M and 1A=0.18M for E_2. In our case, M/m=0.91 and m$_s$/m=1.11. These values were obtained by benchmarking full implementations on a 3.4GHz Intel Core i7-2600 (Sandy Bridge) processor.

Curve	Method	Protection	# Cores	Operation Count	Total Cost
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	no	1	1i + 742m + 225s + 767a	1117m
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 36pts.	yes	1	1i + 1014m$_s$ + 217s$_s$ + 997a + 68t	1525m$_s$≈1693m
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	no	4	1i + 420m + 198s + 484a	724m
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 20pts.	yes	4	1i + 503m$_s$ + 196s$_s$ + 532a + 22t	848m$_s$≈941m
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 32pts.	no	1	2i + 617m + 404s + 847a	1209m
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 36pts.	yes	1	2i + 849m$_s$ + 489s$_s$ + 1001a + 68t	1630m$_s$≈1809m
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 32pts.	no	4	2i + 371m + 316s + 593a	850m
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 36pts.	yes	4	2i + 425m$_s$ + 335s$_s$ + 637a + 17t	977m$_s$≈1084m
$E'_1(\mathbb{F}_{p_1^2})$	non-GLV, 8pts.	no	1	2i + 1169m + 1169s + 2141a	2575m
$E_2(\mathbb{F}_{p_2})$	2-GLV, 16pts.	no	1	1I + 904M + 690S + 1240A	2004M≈1824m
$E_2(\mathbb{F}_{p_2})$	2-GLV, 16pts.	no	2	1I + 681M + 615S + 1103A	1692M≈1540m

Focusing on curve E'_1, it can be noted a significant cost reduction when switching from non-GLV to a GLV-GLS implementation. The speedup is more than twofold for sequential, unprotected versions. Significant improvements are also expected when using multiple cores. A remarkable factor 3 speedup is expected when using GLV-GLS on four cores in comparison with a traditional execution (listed as non-GLV).

Table 3. Point multiplication timings (in clock cycles), 64-bit processor

Curve	Method	Protection	# Cores	Core i7
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	no	1	**91,000**
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 36pts.	yes	1	137,000
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 16pts.	no	4	**61,000**
$E'_{T3}(\mathbb{F}_{p_3^2})$	4-GLV-GLS, 20pts.	yes	4	78,000
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 32pts.	no	1	99,000
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 36pts.	yes	1	145,000
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 32pts.	no	4	70,000
$E'_1(\mathbb{F}_{p_1^2})$	4-GLV-GLS, 36pts.	yes	4	89,000
$E'_1(\mathbb{F}_{p_1^2})$	non-GLV, 8pts.	no	1	201,000
$E_2(\mathbb{F}_{p_2})$	2-GLV, 16pts.	no	1	151,000
$E_2(\mathbb{F}_{p_2})$	2-GLV, 16pts.	no	2	127,000

In general for our targeted GLV-GLS curves, the speedup obtained by using four cores is in between 42%-80%. Interestingly, the improvement is greater for protected implementations since the overhead of using a regular pattern execution is minimized when distributing computation among various cores. Remarkably, protecting implementations against timing attacks increases cost in between 28%-52%, approximately. On the other hand, in comparison with curve E_2, an optimal execution of GLV-GLS on four cores is expected to run 1.81 faster than an optimal execution of the standard 2-GLV on two cores.

To confirm our findings we implemented the different versions using curves E'_1, E_2 and E'_{T3}. To achieve maximum performance and ease the task of parallelizing and protecting the implementations, we wrote our own standalone software without employing any external library. For our experiments we used a 3.4GHz Intel Core i7-2600 processor, which contains four cores. The timings in terms of clock cycles are displayed in Table 3. As can be seen, closely following our analysis GLV-GLS achieves a twofold speedup over a non-GLV implementation on a single core. Parallel execution improves performance by up to 76% for side-channel protected versions. In comparison with the non-GLV implementation, the four-core implementation runs 3 times faster. Our results also confirm the lower-than-expected cost of adding side-channel protection. Sequential versions lose about 50% in performance whereas parallel versions only lose about 28%. The relative speedup when moving from 2-GLV to GLV-GLS on j-invariant 0 curves is 53%, closely following the theoretical 50% estimated previously. Four-core GLV-GLS supports a computation that runs 81% faster than the standard 2-GLV on two cores. Finally, in practice our Twisted Edwards curve achieves up to 9% speedup on the sequential, non-protected scenario in comparison with the efficient j-invariant 0 curve based on Jacobian coordinates.

Let us now compare our best numbers with recent results in the literature. Focusing on one-core unprotected implementations, the first author together with Hu and Xu reported in [13] 122,000 cycles for a j-invariant 0 Weierstrass curve on an Intel Core i7-2600 processor. We report 91,000 cycles with the GLV-GLS Twisted Edwards curve E'_{T3}, improving that number in 34%. We benchmarked on the same processor the side-channel protected software recently presented by Bernstein et al. in [3], and obtained 194,000 cycles. Thus, our protected implementation, which runs in 137,000 cycles, improves that result in 42%. Our result is also 12% faster than the recent implementation by Hamburg [11]. Recent implementations on multiple cores are reported by Taverne et al. in [27].

However, they do not explore the 128-bit security level in their implementations and, hence, results are not directly comparable. They also report a protected implementation of a binary Edwards curve that runs in 225,000 cycles on a Core i7-2600 machine, which is 64% slower than our corresponding result. Since the advent of the carryless multiplier on recent Intel processors, it has been suspected that the only curves able to get performance as good as the GLV-GLS method over large prime characteristic fields are Koblitz curves over binary fields. In fact, Aranha et al. [1] very recently presented an implementation of the Koblitz curve K-283 that runs in 99,000 cycles on an Intel Core i7-2600, which is 9% slower than our GLV-GLS Twisted Edwards curve E'_{T3} (unprotected sequential execution). We remark that such performance for a binary elliptic curve can only be attained on very recent processors that possess the so-called carryless multiplier. Aranha et al. do not report timings for side-channel protected implementations. To the best of our knowledge, we have presented the first scalar multiplication implementation running on multiple cores that is protected against timing attacks, cache attacks and several others.

8 Conclusion

We have shown how to generalize the GLV scalar multiplication method by combining it with Galbraith-Lin-Scott's ideas to perform a proven almost fourfold speedup on GLV curves over $\mathbb{F}_{p^2}$. We have introduced a new and easy-to-implement reduction algorithm, consisting in two applications of the extended Euclidean algorithm, one in $\mathbb{Z}$ and the other in $\mathbb{Z}[i]$. The refined bound obtained from this algorithm has allowed us to get a relative improvement from 2-GLV to 4-GLV-GLS *quasi-independent* of the curve. Our analysis and experimental results on different GLV curves show that in practice one should expect speedups close to 50%. We improve performance even further by exploiting the Twisted Edwards model over a larger set of curves and show that this approach is especially significant to certain GLV curves with slow arithmetic in the Weierstrass model. This makes available to implementers new curves that achieve close to optimal performance. Moreover, we have shown how to protect GLV-based implementations against certain side-channel attacks with relatively low overhead and carried out a performance analysis on modern multicore processors. Our implementations of the GLV-GLS method improve the state-of-the-art performance of point multiplication for multiple scenarios: unprotected and side-channel protected versions with sequential and parallel execution.

Acknowledgements: We thank the reviewers and Mike Scott for their helpful comments. Also, we would like to thank Diego Aranha for his advice on multicore programming, Joppe Bos for his help on looking for efficient chains for implementing modular inversion, Craig Costello and Kristin Lauter for helping us to detect a typo on a curve parameter in a previous paper version.

References

1. D.F. Aranha, A. Faz-Hernandez, J. Lopez, and F. Rodriguez-Henriquez. Faster implementation of scalar multiplication on Koblitz curves. In *Proceedings of Latincrypt 2012*, volume 7533 of *LNCS*, pages 177–193. Springer, 2012.
2. D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted edwards curves. In S. Vaudenay, editor, *Proceedings of AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 389–405. Springer, 2008.
3. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. In B. Preneel and T. Takagi, editors, *Proceedings of CHES 2011*, volume 6917 of *LNCS*, pages 124–142. Springer, 2011.

4. D. Brumley and D. Boneh. Remote timing attacks are practical. In S. Mangard and F.-X. Standaert, editors, *Proceedings of the 12th USENIX Security Symposium*, volume 6225 of *LNCS*, pages 80–94. Springer, 2003.
5. H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer, 1996.
6. G. Cornacchia. Su di un metodo per la risoluzione in numeri interi dell'equazione $\sum_{h=0}^{n} C_h x^{n-h} y^h = P$. *Giornale di Mathematiche di Battaglini*, 46:33–90, 1908.
7. H. Edwards. A normal form for elliptic curves. In *Bulletin of the American Mathematical Society*, volume 44, pages 393–422, 2007.
8. S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. In A. Joux, editor, *Proceedings of EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 518–535. Springer, 2009.
9. S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. In *J. Cryptology*, volume 24(3), pages 446–469, 2011.
10. 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 *LNCS*, pages 190–200. Springer, 2001.
11. M. Hamburg. Fast and compact elliptic-curve cryptography. In *Cryptology ePrint Archive, Report 2012/309*, 2012. Available at: `http://eprint.iacr.org/2012/309`.
12. H. Hisil, K. Wong, G. Carter, and E. Dawson. Twisted edwards curves revisited. In J. Pieprzyk, editor, *Proceedings of ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 326–343. Springer, 2008.
13. Z. Hu, P. Longa, and M. Xu. Implementing 4-dimensional GLV method on GLS elliptic curves with j-invariant 0. *Designs, Codes and Cryptography*, 63(3):331–343, 2012. Also in Cryptology ePrint Archive, Report 2011/315, `http://eprint.iacr.org/2011/315`.
14. M. Joye and M. Tunstall. Exponent recoding and regular exponentiation algorithms. In M. Joye, editor, *Proceedings of Africacrypt 2003*, volume 5580 of *LNCS*, pages 334–349. Springer, 2009.
15. E. Kasper. Fast elliptic curve cryptography in openssl. In *2nd Workshop on Real-Life Cryptographic Protocols and Standardization*, 2011.
16. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - Proceedings of CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
17. A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
18. P. Longa. *High-speed elliptic curve and pairing-based cryptography*. PhD thesis, University of Waterloo, 2011. Available at: `http://hdl.handle.net/10012/5857`.
19. P. Longa and C. Gebotys. Efficient techniques for high-speed elliptic curve cryptography. In S. Mangard and F.-X. Standaert, editors, *Proceedings of CHES 2010*, volume 6225 of *LNCS*, pages 80–94. Springer, 2010.
20. P. Longa and A. Miri. New composite operations and precomputation scheme for elliptic curve cryptosystems over prime fields. In R. Cramer, editor, *Proceedings of PKC 2008*, volume 4939 of *LNCS*, pages 229–247. Springer, 2008.
21. P. Longa and F. Sica. Four-dimensional Gallant-Lambert-Vanstone scalar multiplication (full version). In *Cryptology ePrint Archive, Report 2011/608*, 2012. Available at: `http://eprint.iacr.org/2011/608`.
22. F. Morain. *Courbes elliptiques et tests de primalité*. PhD thesis, Université de Lyon I, Available at: `http://www.lix.polytechnique.fr/\homedirmorain/Articles/\linebreakarticles.english.html`, 1990. Chapter 2: On Cornacchia's algorithm (joint with J-L. Nicolas).
23. P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. In Duncan A. Buell, editor, *Algorithmic Number Theory, 6th International Symposium, ANTS-VI, Burlington, VT, USA, June 13-18, 2004, Proceedings*, volume 3076 of *LNCS*, pages 338–357. Springer, 2004.
24. K. Okeya and T. Takagi. The width-w naf method provides small memory and fast elliptic curve scalars multiplications against side-channel attacks. In M. Joye, editor, *Proceedings of CT-RSA 2003*, volume 2612 of *LNCS*, pages 328–342. Springer, 2003.
25. 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, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002*, volume 2595 of *LNCS*, pages 21–36. Springer, 2002.
26. H. M. Stark. Class-numbers of complex quadratic fields. In *Modular functions of one variable, I (Proc. Internat. Summer School, Univ. Antwerp, Antwerp, 1972)*, pages 153–174. Lecture Notes in Mathematics, Vol. 320. Springer, Berlin, 1973.

27. J. Taverne, A. Faz-Hernandez, D.F. Aranha, F. Rodriguez-Henriquez, D. Hankerson, and J. Lopez. Software implementation of binary elliptic curves: impact of the carry-less multiplier on scalar multiplication. In B. Preneel and T. Takagi, editors, *Proceedings of CHES 2011*, volume 6917 of *LNCS*, pages 108–123. Springer, 2011.
28. Z. Zhou, Z. Hu, M. Xu, and W. Song. Efficient 3-dimensional GLV method for faster point multiplication on some GLS elliptic curves. *Inf. Proc. Lett.*, 77(262):1075–1104, 2010.

A Examples

We give a few examples of GLV curves, which are curves defined over $\mathbb{C}$ with complex multiplication by a quadratic integer of small norm, corresponding to an endomorphism ϕ of small degree[6]. They make up an exhaustive list, up to isomorphism, in increasing order of endomorphism degree up to degree 3. While the first four examples appear in the previous literature, the next ones (degree 3) are new and have been computed with the Stark algorithm [26].

Example 1. Let $p \equiv 1 \pmod 4$ be a prime. Define an elliptic curve E over $\mathbb{F}_p$ by

$$y^2 = x^3 + ax \ .$$

If β is an element of order 4, then the map ϕ defined in the affine plane by

$$\phi(x, y) = (-x, \beta y)$$

is an endomorphism of E defined over $\mathbb{F}_p$ with $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\sqrt{-1}]$, since ϕ satisfies the equation[7]

$$\phi^2 + 1 = 0 \ .$$

Example 2. Let $p \equiv 1 \pmod 3$ be a prime. Define an elliptic curve E over $\mathbb{F}_p$ by

$$y^2 = x^3 + b \ .$$

If γ is an element of order 3, then we have an endomorphism ϕ defined over $\mathbb{F}_p$ by

$$\phi(x, y) = (\gamma x, y) \ ,$$

and $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\frac{1+\sqrt{-3}}{2}]$, since ϕ satisfies the equation

$$\phi^2 + \phi + 1 = 0 \ .$$

Example 3. Let $p > 3$ be a prime such that -7 is a quadratic residue modulo p. Define an elliptic curve E over $\mathbb{F}_p$ by

$$y^2 = x^3 - \frac{3}{4}x^2 - 2x - 1 \ .$$

If $\xi = (1 + \sqrt{-7})/2$ and $a = (\xi - 3)/4$, then we get the $\mathbb{F}_p$-endomorphism ϕ defined by

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

and $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\frac{1+\sqrt{-7}}{2}]$, since ϕ satisfies the equation

$$\phi^2 - \phi + 2 = 0 \ .$$

[6] By small we mean really small, usually less than 5. In particular, for cryptographic applications, the degree is much smaller than the field size.
[7] This is the only case when we cannot apply Lemma 1. It needs a separate treatment, given in [21], Appendix B.

Example 4. Let $p > 3$ be a prime such that -2 is a quadratic residue modulo p. Define an elliptic curve E over $\mathbb{F}_p$ by

$$y^2 = 4x^3 - 30x - 28$$

together with the $\mathbb{F}_p$-endomorphism ϕ defined[8] by

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

We have $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\sqrt{-2}]$ since ϕ satisfies the equation

$$\phi^2 + 2 = 0 \ .$$

Example 5. Let $p > 3$ be a prime such that -11 is a quadratic residue modulo p. We define the elliptic curve E over $\mathbb{F}_p$

$$y^2 = x^3 - \frac{13824}{539}x + \frac{27648}{539}$$

with $a = (1+\sqrt{-11})/2$ and the endomorphism ϕ defined by

$$\phi(x,y) = \left(\frac{\left(-\frac{539}{5184}a + \frac{539}{1728}\right)x^3 + \left(\frac{28}{27}a - \frac{35}{18}\right)x^2 + \left(-\frac{92}{9}a + \frac{8}{3}\right)x + \frac{1728}{77}a + \frac{192}{77}}{\left(\frac{2695}{5184}a - \frac{539}{864}\right)x^2 + \left(-\frac{217}{54}a + \frac{49}{18}\right)x + \frac{64}{9}a - \frac{4}{3}}, y \right.$$
$$\left. \frac{\left(\frac{3773}{373248}a - \frac{18865}{995328}\right)x^3 + \left(-\frac{2695}{20736}a + \frac{539}{3456}\right)x^2 + \left(\frac{7}{432}a - \frac{91}{144}\right)x + \frac{20}{27}a + \frac{1}{9}}{\left(-\frac{18865}{1492992}a + \frac{116963}{995328}\right)x^3 + \left(\frac{7007}{20736}a - \frac{539}{432}\right)x^2 + \left(-\frac{791}{432}a + \frac{581}{144}\right)x + \frac{74}{27}a - \frac{35}{9}} \right)$$

such that $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\frac{1+\sqrt{-11}}{2}]$. The characteristic polynomial of ϕ is

$$\phi^2 - \phi + 3 = 0 \ .$$

Example 6. Let $p > 3$ be a prime such that -3 is a quadratic residue mod p. We define the elliptic curve E over $\mathbb{F}_p$

$$y^2 = x^3 - \frac{3375}{121}x + \frac{6750}{121}$$

with the endomorphism ϕ defined by

$$\phi(x,y) = \left(-\frac{1331x^3 - 10890x^2 + 81675x - 189000}{33(11x-45)^2}, y\,\frac{1331x^3 - 16335x^2 + 7425x + 43875}{3\sqrt{-3}(11x-45)^3} \right)$$

such that[9] $\mathrm{End}(E) = \mathbb{Z}[\phi] \cong \mathbb{Z}[\sqrt{-3}]$. The characteristic polynomial of ϕ is

$$\phi^2 + 3 = 0 \ .$$

[8] We take the opportunity to correct a typo found and transmitted in many sources, where a y factor was absent in the second coordinate. Its sign is irrelevant.

[9] This is the first example where the endomorphism ring is not the maximal order of its field of fractions. It can be summarily seen as follows: $\mathrm{End}(E) \supseteq \mathbb{Z}[\sqrt{-3}]$. If not equal, then it must be the full ring of integers $\mathbb{Z}[\frac{1+\sqrt{-3}}{2}]$. This would imply that $j = 0$, as there is only $h(-3) = 1$ isomorphism class of elliptic curves with complex multiplication by $\mathbb{Z}[\frac{1+\sqrt{-3}}{2}]$, given in Example 2 (see [26] for an abridged description of the theory of complex multiplication). This is clearly not the case here. Alternatively, one can see that there would exist a nontrivial automorphism (a primitive cube root of unity) corresponding to $\frac{-1+\sqrt{-3}}{2}$. A direct computation then shows this is impossible.

B A New Four-Dimensional Lattice Reduction Algorithm

Algorithm 1 (Cornacchia's GCD algorithm in $\mathbb{Z}$)

Input: $n \equiv 1 \pmod 4$ prime, $1 < \mu < n$ such that $\mu^2 \equiv -1 \pmod n$.
Output: $\nu = \nu_{(R)} + i\nu_{(I)}$ Gaussian prime dividing n, such that $\nu P = 0$.

---

1. **initialize:**
   $r_0 \leftarrow n,\ r_1 \leftarrow \mu,\ r_2 \leftarrow n,$
   $t_0 \leftarrow 0,\ t_1 \leftarrow 1,\ t_2 \leftarrow 0,$
   $q \leftarrow 0.$
2. **main loop:**
   while $r_2^2 \geq n$ do
      $q \leftarrow \lfloor r_0/r_1 \rfloor,$
      $r_2 \leftarrow r_0 - qr_1,\ r_0 \leftarrow r_1,\ r_1 \leftarrow r_2,$
      $t_2 \leftarrow t_0 - qt_1,\ t_0 \leftarrow t_1,\ t_1 \leftarrow t_2.$
3. **return:**
   $\nu = r_1 - it_1,\ \nu_{(R)} = r_1,\ \nu_{(I)} = -t_1$

Algorithm 2 (Cornacchia's algorithm in $\mathbb{Z}[i]$ - compact form)

Input: ν Gaussian prime dividing n rational prime, $1 < \lambda < n$ such that $\lambda^2 + r\lambda + s \equiv 0 \pmod n$.
Output: Two $\mathbb{Z}[i]$-linearly independent vectors v_1 & v_2 of $\ker F \subset \mathbb{Z}[i]^2$ of rectangle norms $< 51.5(\sqrt{1 + |r| + s})\, n^{1/4}$.

---

1. **initialize:**
   If $\lambda^2 \geq 2n$ then
      $r_0 \leftarrow \lambda,$
   else
      $r_0 \leftarrow \lambda + n,$
   $r_1 \leftarrow \nu,\ r_2 \leftarrow n,$
   $s_0 \leftarrow 1,\ s_1 \leftarrow 0,\ s_2 \leftarrow 0,$
   $q \leftarrow 0.$
2. **main loop:**
   while $|r_2|^4(1 + |r| + s)^2 \geq n$ do
      $q \leftarrow$ closest Gaussian integer to $r_0/r_1,$
      $r_2 \leftarrow r_0 - qr_1,\ r_0 \leftarrow r_1,\ r_1 \leftarrow r_2,$
      $s_2 \leftarrow s_0 - qs_1,\ s_0 \leftarrow s_1,\ s_1 \leftarrow s_2.$
3. **return:**
   $v_1 = (r_0, -s_0),\ v_2 = (r_1, -s_1)$
