# Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves

Steven D. Galbraith[1,⋆], Xibin Lin[2,⋆⋆], and Michael Scott[3,⋆⋆⋆]

[1] Mathematics Department,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX,
United Kingdom
steven.galbraith@rhul.ac.uk
[2] School of Mathematics and Computational Science,
Sun Yat-Sen University, Guangzhou, 510275, P.R. China
linxibin@mail2.sysu.edu.cn
[3] School of Computing, Dublin City University,
Ballymun, Dublin 9, Ireland
mike@computing.dcu.ie

**Abstract.** Efficiently computable homomorphisms allow elliptic curve point multiplication to be accelerated using the Gallant-Lambert-Vanstone (GLV) method. We extend results of Iijima, Matsuo, Chao and Tsujii which give such homomorphisms for a large class of elliptic curves by working over $\mathbb{F}_{p^2}$ and demonstrate that these results can be applied to the GLV method.

In general we expect our method to require about 0.75 the time of previous best methods (except for subfield curves, for which Frobenius expansions can be used). We give detailed implementation results which show that the method runs in between 0.70 and 0.84 the time of the previous best methods for elliptic curve point multiplication on general curves.

**Keywords:** elliptic curves, point multiplication, GLV method, isogenies.

## 1 Introduction

Let $E$ be an elliptic curve over a finite field $\mathbb{F}_q$ and let $P, Q \in E(\mathbb{F}_q)$ have order $r$. The fundamental operations in elliptic curve cryptography are point multiplication $[n]P$ and multiexponentiation $[n]P + [m]Q$ where $n, m \in \mathbb{Z}$. There is a vast literature on efficient methods for computing $[n]P$ and $[n]P + [m]Q$ (a good reference is [3]). There is a significant difference between computing $[n]P$ for varying $n$ and a fixed point $P$, and computing $[n]P$ where both $n$ and $P$ vary; this paper focusses on the latter case.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

[⋆] This work supported by EPSRC grant EP/D069904/1.

[⋆⋆] This author thanks the Chinese Scholarship Council.

[⋆⋆⋆] This author acknowledges support from the Science Foundation Ireland under Grant No. 06/MI/006.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 518–535, 2009.

The Gallant-Lambert-Vanstone (GLV) method [15] is an important tool for speeding up point multiplication. The basic idea is as follows. If the elliptic curve $E$ has an efficiently computable endomorphism $\psi$ (other than a standard multiplication by $n$ map) such that $\psi(P) \in \langle P \rangle$ then one can replace the computation $[n]P$ by the multiexponentiation $[n_0]P + [n_1]\psi(P)$ where $|n_0|, |n_1| \approx \sqrt{r}$. The integers $n_0$ and $n_1$ are computed by solving a closest vector problem in a lattice, see [15] for details. In principle this computation requires only around 0.6 to 0.7 the time of the previous method (the precise details depend on the relative costs of doubling and addition, the window size being used, etc). Some examples allow higher degree decompositions such as $[n_0] + [n_1]\psi(P) + \cdots + [n_{m-1}]\psi^{m-1}(P)$ where $|n_i| \approx r^{1/m}$ which can give further speedups. We call the latter approach the $m$-dimensional GLV method.

Gallant, Lambert and Vanstone [15] only gave examples of suitable efficiently computable endomorphisms in two cases, namely subfield curves (i.e., groups $E(\mathbb{F}_{q^m})$ where $E$ is defined over $\mathbb{F}_q$; these do not have prime or nearly prime order unless $q$ is very small) and curves with special endomorphism structure (essentially, that the endomorphism ring has small class number). Hence, if one is using randomly chosen prime-order elliptic curves over finite fields for cryptography (or if one wants to use special primes such as NIST primes, see Section 2.2.6 of [18]) then the GLV method is not usually available. Indeed, in Section 7 of [33] one finds the claim "the GLV method is only effective for those exceptional elliptic curves that have complex multiplication by an order with small discriminant."

In fact, Iijima, Matsuo, Chao and Tsujii [20] constructed an efficiently computable homomorphism on elliptic curves $E(\mathbb{F}_{p^2})$ with $j(E) \in \mathbb{F}_p$ arising from the Frobenius map on a twist of $E$. Apparently they did not realise the application of their results to the GLV method. In this paper we give a generalisation of the Iijima-Matsuo-Chao-Tsujii (IMCT) construction and analyse it in the context of the GLV method. The construction applies to all elliptic curves over $\mathbb{F}_{p^2}$ such that $j(E) \in \mathbb{F}_p$ and, as noted in [20,29], can be used with curves of prime order.

The curves considered in this paper are not completely general: the number of $\mathbb{F}_{q^2}$-isogeny classes of elliptic curves over $\mathbb{F}_{q^2}$ is approximately $2q^2$ whereas the construction in Section 2 gives only approximately $q$ isomorphism classes of curves. However, this is a major improvement over earlier papers on the GLV method which, in practice, were only applied to a finite number of $\mathbb{F}_q$-isomorphism classes for any given $q$. The results of this paper therefore overturn the claims of Section 7 of [33].

The basic idea is somewhat analogous to subfield curves: We take elliptic curves $E$ with $j(E) \in \mathbb{F}_q$ and consider the group $E(\mathbb{F}_{q^m})$. However a crucial difference is that $E$ is defined over $\mathbb{F}_{q^m}$, not $\mathbb{F}_q$. This means that it is possible to obtain curves of prime order and so there is no need to restrict attention to $q$ being small. Our method can be used with any prime power $q$ and any elliptic curves $E$ over $\mathbb{F}_q$ and always gives rise to a GLV method of dimension at least two.

We give experimental results comparing the cost of our algorithm for point multiplication $[n](x, y)$ with previous methods for this operation (indeed, we compare with optimised implementations due to Bernstein [4] and Gaudry-Thomé [17], which, based on ideas of Montgomery [28], use only $x$-coordinate arithmetic). The purpose of our implementation experiments is to give a good picture of the speedup obtained with the new method compared with using curves over prime fields; we stress that our implementation is not claimed to be the best possible and that one could probably achieve further speedups from a different choice of curve coordinates or different exponentiation methods.

We find that the new method runs in between 0.70 and 0.84 the time of the previous best methods. The exact performance depends on the platform being used; our best result is for 8-bit processors. Our methods (unlike methods using Montgomery ladders, such as [4,17]) can also be used for signature verification. Our experimental results in Table 4 show that Schnorr signature verification runs in around 0.73 the time of the best previous methods for the same curve.

Note that our techniques can also be implemented on elliptic curves given by any equation (e.g., Edwards or Jacobi-quartic form, see [6,7,8]) and exploit their benefits. We also generalise the method to hyperelliptic curves. The details of both these cases are omitted due to lack of space, but are given in the full version of the paper.

The focus in this paper is on curves over fields of large prime characteristic, since in small characteristic one might prefer to use subfield curves and Frobenius expansions. However, Hankerson, Karabina and Menezes [19] have experimented with the method in characteristic 2 and they report that the new method runs in about 0.74 to 0.77 the time of the best standard method for general curves.

We now give an outline of the paper. First we describe the homomorphism and explain how it leads to a 2-dimensional GLV method. Section 3 gives a specific key generation algorithm which may be convenient for some applications. Section 4 shows how to get a 4-dimensional GLV method for $y^2 = x^3 + B$ over $\mathbb{F}_{p^2}$. Section 5 gives some details about our implementation. The proof of the pudding is the timings in Section 6. Section 7 discusses known security threats from using the construction and explains how to avoid them.

## 2 The Homomorphism

We consider elliptic curves defined over any field $\mathbb{F}_q$ with identity point $\mathcal{O}_E$. Recall that if $E$ is an elliptic curve over $\mathbb{F}_q$ with $q + 1 - t$ points then one can compute the number of points $\#E(\mathbb{F}_{q^m})$ efficiently. For example, $\#E(\mathbb{F}_{q^2}) = q^2 + 1 - (t^2 - 2q) = (q + 1)^2 - t^2$. As usual we define

$$E(\mathbb{F}_{q^m})[r] = \{P \in E(\mathbb{F}_{q^m}) : [r]P = \mathcal{O}_E\}.$$

When we say that a curve or mapping is 'defined over $\mathbb{F}_{q^k}$' we mean that the coefficients of the polynomials are all in $\mathbb{F}_{q^k}$. The implicit assumption throughout the paper is that when we say an object is defined over a field $\mathbb{F}_{q^k}$ then it is not defined over any smaller field, unless explicitly mentioned.

The following result gives the main construction. Novices can replace the words 'separable isogeny' with 'isomorphism', set $d = 1$ and replace $\hat{\phi}$ by $\phi^{-1}$

without any significant loss of functionality (in which case one essentially obtains the result of Iijima et al [20]). Recall that if $r$ is a prime we write $r \| N$ to mean $r \mid N$ but $r^2 \nmid N$.

**Theorem 1.** *Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ such that $\#E(\mathbb{F}_q) = q+1-t$ and let $\phi : E \rightarrow E'$ be a separable isogeny of degree $d$ defined over $\mathbb{F}_{q^k}$ where $E'$ is an elliptic curve defined over $\mathbb{F}_{q^m}$ with $m \mid k$. Let $r \mid \#E'(\mathbb{F}_{q^m})$ be a prime such that $r > d$ and such that $r \| \#E'(\mathbb{F}_{q^k})$. Let $\pi$ be the $q$-power Frobenius map on $E$ and let $\hat{\phi} : E' \rightarrow E$ be the dual isogeny of $\phi$. Define*

$$\psi = \phi\pi\hat{\phi}.$$

*Then*

1. *$\psi \in End_{\mathbb{F}_{q^k}}(E')$ (i.e., $\psi$ is a group homomorphism).*
2. *For all $P \in E'(\mathbb{F}_{q^k})$ we have $\psi^k(P) - [d^k]P = \mathcal{O}_E$ and $\psi^2(P) - [dt]\psi(P) + [d^2q]P = \mathcal{O}_E$.*
3. *There is some $\lambda \in \mathbb{Z}$ such that $\lambda^k - d^k \equiv 0 \pmod{r}$ and $\lambda^2 - dt\lambda + d^2q \equiv 0 \pmod{r}$ such that $\psi(P) = [\lambda]P$ for all $P \in E'(\mathbb{F}_{q^m})[r]$.*

*Proof.* First note that $\hat{\phi}$ is an isogeny from $E'$ to $E$ and is defined over $\mathbb{F}_{q^k}$, that $\pi$ is an isogeny from $E$ to itself defined over $\mathbb{F}_q$, and that $\phi$ is an isogeny from $E$ to $E'$ defined over $\mathbb{F}_{q^k}$. Hence $\psi$ is an isogeny of $E'$ to itself, and is defined over $\mathbb{F}_{q^k}$ (or maybe a subfield). Therefore, $\psi$ is a group homomorphism.

Since $\phi\hat{\phi} = d$ on $E'$ it follows that

$$\psi^2 = \phi\pi\hat{\phi}\phi\pi\hat{\phi} = \phi\pi d\pi\hat{\phi} = d\phi\pi^2\hat{\phi}$$

and, by induction, $\psi^k = d^{k-1}\phi\pi^k\hat{\phi}$. For $P \in E'(\mathbb{F}_{q^k})$ we have $\hat{\phi}(P) \in E(\mathbb{F}_{q^k})$ and so $\pi^k(\hat{\phi}(P)) = \hat{\phi}(P)$. Hence $\psi^k(P) = [d^k]P$.

Similarly, writing $Q = \hat{\phi}(P)$ for $P \in E'(\mathbb{F}_{q^k})$ we have $\pi^2(Q) - [t]\pi(Q) + [q]Q = \mathcal{O}_E$ and so $[d]\phi(\pi^2 - [t]\pi + [q])\hat{\phi}(P) = \mathcal{O}_E$. Using the previous algebra, this implies

$$(\psi^2 - [dt]\psi + [qd^2])P = \mathcal{O}_E.$$

Finally, let $P \in E'(\mathbb{F}_{q^m})$ have order $r$. Since $\psi(P) \in E'(\mathbb{F}_{q^k})$ also has order $r$ and $r \| \#E'(\mathbb{F}_{q^k})$ it follows that $\psi(P) = [\lambda]P$ for some $\lambda \in \mathbb{Z}$. Since $\psi$ is a homomorphism, $\psi([a]P) = [a]\psi(P) = [\lambda]([a]P)$ for all $a \in \mathbb{Z}$. Since $\psi^k(P) - [d^k]P = [\lambda^k]P - [d^k]P = \mathcal{O}_E$ it follows that $\lambda^k - d^k \equiv 0 \pmod{r}$. Similarly, $\lambda^2 - dt\lambda + d^2q \equiv 0 \pmod{r}$. □

We stress that there is nothing unexpected in the above construction. Consider the case when $\phi$ is an isomorphism: Then $E' \cong E$ implies $\mathrm{End}(E') \cong \mathrm{End}(E)$. We know that $\mathrm{End}(E)$ contains the $p$-power Frobenius map and hence $\mathrm{End}(E')$ contains a corresponding endomorphism. The above Theorem simply writes down this endomorphism explicitly.

The proof generalises immediately to hyperelliptic curves (see the full version of this paper or [22]).

### 2.1 Special Case of Quadratic Twists

We now specialise Theorem 1 to elliptic curves over $\mathbb{F}_p$ where $p > 3$ and the case $m = 2$.

**Corollary 1.** *Let $p > 3$ be a prime and let $E$ be an elliptic curve over $\mathbb{F}_p$ with $p + 1 - t$ points. Let $E'$ over $\mathbb{F}_{p^2}$ be the quadratic twist of $E(\mathbb{F}_{p^2})$. Then $\#E'(\mathbb{F}_{p^2}) = (p-1)^2 + t^2$. Let $\phi : E \to E'$ be the twisting isomorphism defined over $\mathbb{F}_{p^4}$. Let $r \mid \#E'(\mathbb{F}_{p^2})$ be a prime such that $r > 2p$ Let $\psi = \phi\pi\phi^{-1}$. For $P \in E'(\mathbb{F}_{p^2})[r]$ we have $\psi^2(P) + P = \mathcal{O}_E$.*

*Proof.* Let $E : y^2 = x^3 + Ax + B$ with $A, B \in \mathbb{F}_p$. We have $\#E(\mathbb{F}_{p^2}) = p^2 + 1 - (t^2 - 2p)$. Let $u \in \mathbb{F}_{p^2}$ be a non-square in $\mathbb{F}_{p^2}$, define $A' = u^2 A, B' = u^3 B$ and $E' : y^2 = x^3 + A'x + B'$. Then $E'$ is the quadratic twist of $E(\mathbb{F}_{p^2})$ and $\#E'(\mathbb{F}_{p^2}) = p^2 + 1 + (t^2 - 2p) = (p-1)^2 + t^2$. The isomorphism $\phi : E \to E'$ is given by

$$\phi(x, y) = (ux, \sqrt{u}^3 y)$$

and is defined over $\mathbb{F}_{p^4}$.

If $r \mid \#E'(\mathbb{F}_{p^2})$ is prime such that $r > 2p$ then $r \nmid \#E(\mathbb{F}_{p^2}) = (p+1-t)(p+1+t)$ and so $r \| \#E'(\mathbb{F}_{p^4}) = \#E(\mathbb{F}_{p^2})\#E'(\mathbb{F}_{p^2})$. Hence we may apply Theorem 1. This shows that $\psi = \phi\pi\phi^{-1}$ is a group homomorphism such that $\psi(P) = [\lambda]P$ for $P \in E'(\mathbb{F}_{p^2})[r]$ where $\lambda^4 - 1 \equiv 0 \pmod{r}$. We now show that, in fact, $\lambda^2 + 1 \equiv 0 \pmod{r}$.

By definition, $\psi(x, y) = (ux^p/u^p, \sqrt{u}^3 y^p/\sqrt{u}^{3p})$ where $u \in \mathbb{F}_{p^2}$ (i.e., $u^{p^2} = u$) and $\sqrt{u} \notin \mathbb{F}_{p^2}$ (and so, $\sqrt{u}^{p^2} = -\sqrt{u}$). If $P = (x, y) \in E'(\mathbb{F}_{p^2})$ then $x^{p^2} = x, y^{p^2} = y$ and so

$$\begin{aligned}\psi^2(x, y) &= (ux^{p^2}/u^{p^2}, \sqrt{u}^3 y^{p^2}/\sqrt{u}^{3p^2}) \\ &= (x, (-1)^3 y) \\ &= -(x, y).\end{aligned}$$

This completes the proof. □

The above result applies to any elliptic curve over $\mathbb{F}_p$ (with $p > 3$) and shows that the 2-dimensional GLV method can be applied. Note that it is possible for $\#E'(\mathbb{F}_{p^2})$ to be prime, since $E'$ is not defined over $\mathbb{F}_p$ (for further analysis see Nogami and Morikawa [29]). One feature of this construction is that, since $p$ is now half the size compared with using elliptic curves over prime fields, point counting is much faster than usual (this was noted in [29]). Since we are dealing with elliptic curves over $\mathbb{F}_{p^2}$, where $p$ is prime, Weil descent attacks are not a threat (see Section 7).

An exercise for the reader is to show that if $E$ is an elliptic curve over $\mathbb{F}_p$ and if $E'$ over $\mathbb{F}_p$ is the quadratic twist of $E$ then the map $\psi$ satisfies $\psi(P) = -P$ for all $P \in E'(\mathbb{F}_p)$. The homomorphism is therefore useless for the GLV method in this case.

**Lemma 1.** *Let $p \equiv 5 \pmod 8$ be a prime. Let notation be as in Corollary 1. Then one may choose*

$$\psi(x, y) = (-x^p, iy^p)$$

*where $i \in \mathbb{F}_p$ satisfies $i^2 = -1$.*

*Proof.* See the full version of the paper. □

**Lemma 2.** *Let notation be as in Corollary 1. Then $\psi(P) = [\lambda]P$ where $\lambda = t^{-1}(p-1) \pmod r$.*

*Proof.* The proof of Corollary 1 shows that $\psi(P) = [\lambda]P$ for some $\lambda \in \mathbb{Z}$. Since $\psi^2(P) = -P$ we have $\lambda^2+1 \equiv 0 \pmod r$. Similarly, $\psi^2(P)-[t]\psi(P)+[p]P = \mathcal{O}_E$, so $\lambda^2 - t\lambda + p \equiv 0 \pmod r$. Subtracting the second equation from the first gives $t\lambda + (1-p) \equiv 0 \pmod r$. □

Finally, we give some remarks about the lattice which arises in the GLV method when decomposing $[n]P$ as $[n_0]P + [n_1]\psi(P)$. Recall from [15] that we consider the lattice

$$L = \{(x, y) \in \mathbb{Z}^2 : x + y\lambda \equiv 0 \pmod r\}.$$

It is easy to prove that $\{(r, 0), (-\lambda, 1)\}$ is a basis for $L$; this shows that the determinant of $L$ is $r$. The GLV method uses Babai's rounding method to solve the closest vector problem (CVP), and this method requires a reduced basis.

**Lemma 3.** *Let notation be as in Corollary 1. The vectors $\{(t, p-1), (1-p, t)\}$ are an orthogonal basis for a sublattice $L'$ of $L$ of determinant $\#E'(\mathbb{F}_{p^2})$. Given a point $(a, b) \in \mathbb{R}^2$ there exists a lattice point $(x, y) \in L'$ such that $\|(a, b)-(x, y)\| \leq (p+1)/\sqrt{2}$.*

*Proof.* By Lemma 2 we have that $t\lambda + (1-p) \equiv 0 \pmod r$, which proves that $(1-p, t) \in L$. Multiplying by $\lambda$ and using $\lambda^2 \equiv -1 \pmod r$ gives $(t, p-1) \in L$. It is easy to check that the vectors are orthogonal and thus linearly independent. The vectors both have length $\sqrt{\#E'(\mathbb{F}_{p^2})} \leq \sqrt{p^2+2p+1} = p+1$. This basis has determinant $(p-1)^2 + t^2 = \#E'(\mathbb{F}_{p^2})$ so generates a sublattice $L' \subseteq L$ (if $\#E'(\mathbb{F}_{p^2}) = r$ then $L = L'$).

Finally, simple geometry shows that the maximum distance from a lattice point is $\sqrt{\#E'(\mathbb{F}_{p^2})/2} \leq (p+1)/\sqrt{2}$. □

Computing the coefficients $n_0, n_1$ for the GLV method is therefore particularly simple in this case (one does not need to use lattice reduction or the methods of [30,21,33]). Further, one knows that $|n_0|, |n_1| \leq (p+1)/\sqrt{2}$. As always, an alternative to the decomposition method which can be used in some cryptographic settings is to choose small coefficients $n_0, n_1 \in \mathbb{Z}$ directly rather than choosing a random $0 \leq n < r$ and then computing the corresponding $(n_0, n_1)$.

### 2.2 Higher Dimension Decompositions

The GLV method can be generalised to $m$-dimensional decompositions $[n]P = [n_0]P + [n_1]\psi(P) + \cdots + [n_{m-1}]\psi^{m-1}(P)$ (for examples with $m = 4$ and $m = 8$ see [13]). Such a setting gives improved performance. As we have found 2-dimensional expansions using $E'(\mathbb{F}_{p^2})$ it is natural to try to get an $m$-dimensional decomposition using $E'(\mathbb{F}_{p^m})$.

In general, to obtain an $m$-dimensional decomposition it is required that $\psi$ does not satisfy any polynomial equation on $E'(\mathbb{F}_{p^m})[r]$ of degree $< m$ with small integer coefficients. Note that $\psi$ always satisfies a quadratic polynomial equation but that the coefficients are not necessarily small modulo $r$.

The following result gives a partial explanation of the behaviour of $\psi$ on $E'(\mathbb{F}_{p^m})$.

**Corollary 2.** *Let $p > 3$ be a prime and let $E$ be an elliptic curve over $\mathbb{F}_p$. Let $E'$ over $\mathbb{F}_{p^m}$ be the quadratic twist of $E(\mathbb{F}_{p^m})$. Write $\phi : E \to E'$ for the twisting isomorphism defined over $\mathbb{F}_{p^{2m}}$. Let $r \mid \#E'(\mathbb{F}_{p^m})$ be a prime such that $r > 2p^{m-1}$ Let $\psi = \phi\pi\phi^{-1}$. For $P \in E'(\mathbb{F}_{p^m})[r]$ we have $\psi^m(P) + P = \mathcal{O}_E$.*

*Proof.* As in Corollary 1, we have $r \| \#E'(\mathbb{F}_{p^{2m}}) = \#E'(\mathbb{F}_{p^m})\#E(\mathbb{F}_{p^m})$ so Theorem 1 applies. Using the same method as the proof of Corollary 1 we have $\psi^m(x, y) = (ux^{p^m}/u^{p^m}, \sqrt{u}^3 y^{p^m}/\sqrt{u}^{3p^m}) = -P$. □

A problem is that the polynomial $x^m + 1$ is not usually irreducible, and it is possible that $\psi$ satisfies a smaller degree polynomial. For example, in the case $m = 3$ one sees that $\#E'(\mathbb{F}_{p^3})$ cannot be prime as it is divisible by $N = \#E(\mathbb{F}_{p^2})/\#E(\mathbb{F}_p)$. If $r \mid \#E'(\mathbb{F}_{p^3})/N$ and $P \in E'(\mathbb{F}_{p^3})[r]$ then $\psi^2(P) - \psi(P) + 1 = \mathcal{O}_E$. Hence one only gets a 2-dimensional decomposition in the case $m = 3$.

Indeed, the interesting case is when $m$ is a power of 2, in which case $x^m + 1$ is irreducible and one can obtain an $m$-dimensional GLV decomposition. Indeed, Nogami and Morikawa [29] already proposed exactly this key generation method (choosing $E$ over $\mathbb{F}_p$ and then using a quadratic twist over $\mathbb{F}_{p^{2^c}}$) as a method to generate curves of prime order. Note that [29] does not consider the GLV method.

Therefore, the next useful case is $m = 4$, giving a 4-dimensional GLV method. On the downside, this case is potentially vulnerable to Weil descent attacks (see Section 7) and so the prime $p$ must be larger than we would ideally like.

The other way to get higher dimension decompositions is to have maps $\phi$ defined over larger fields than a quadratic extension. An example of this is given in Section 4.

## 3 Key Generation

Let $p > 3$ be prime. We present a key generation algorithm for the quadratic twist construction. Our algorithm is designed so that the resulting curve

$E' : y^2 = x^3 + A'x + B'$ over $\mathbb{F}_{p^2}$ has coefficient $A' = -3$, which is convenient for efficient implementation when using Jacobian coordinates (see Section 13.2.1.c of [3] or Section 3.2.2 of [18]). The key generation algorithm can be modified to work with other models for elliptic curves and one can always choose at least one coefficient to have a special form.

We use Lemma 1, which gives a particularly simple map $\psi$. It should be clear that the algorithm can be used in more general cases. Our algorithm produces curves of prime order, but this can be relaxed by requiring only $h < H$ for some bound $H$ in line 7.

**Algorithm 1.** Key generation for quadratic twist construction

```
OUTPUT: $p, E', \psi, \lambda$
 1: Choose a prime $p = 5 \pmod 8$        ▷ e.g., a NIST prime (Section 2.2.6 of [18])
 2: Set $u = \sqrt{2} \in \mathbb{F}_{p^2}$
 3: Set $A' = -3$ and $A = A'/2 \in \mathbb{F}_p$
 4: repeat
 5:     Choose random $B \in \mathbb{F}_p$ and let $E : y^2 = x^3 + Ax + B$
 6:     Compute $t = p + 1 - \#E(\mathbb{F}_p)$.
 7: until $(p-1)^2 + t^2 = hr$ where $r$ is prime and $h = 1$
 8: Set $B' = Bu^3 \in \mathbb{F}_{p^2}$ and $E' : y^2 = x^3 + A'x + B'$
 9: Set $\lambda = t^{-1}(p-1) \pmod r$
10: Compute $i \in \mathbb{F}_p$ so that $i^2 = -1$
11: Define $\psi(x, y) = (-x^p, iy^p)$.
12: return $p, (A', B'), \psi, \lambda$
```

As remarked earlier, key generation is fast compared with standard ECC, since the point counting for $\#E(\mathbb{F}_p)$ is over a field half the usual size (this is precisely the point of the paper [29]).

## 4 Using Special Curves

We have seen that one can obtain a 2-dimensional GLV method for any elliptic curve over $\mathbb{F}_p$. However, 2-dimensional GLV methods were already known for some special curves (i.e., those with a non-trivial automorphism or endomorphism of low degree). We now show how one can get higher-dimensional expansions using elliptic curves $E$ over $\mathbb{F}_{p^2}$ with $\#\mathrm{Aut}(E) > 2$.

The two examples of interest are $E : y^2 = x^3 + B$ and $y^2 = x^3 + Ax$. We give the details in the former case. The latter is analogous.

Let $p \equiv 1 \pmod 6$ and let $B \in \mathbb{F}_p$. Define $E : y^2 = x^3 + B$. Choose $u \in \mathbb{F}_{p^{12}}$ such that $u^6 \in \mathbb{F}_{p^2}$ and define $E' : Y^2 = X^3 + u^6 B$ over $\mathbb{F}_{p^2}$. Repeat the construction (choosing $p, B, u$) until $\#E'(\mathbb{F}_{p^2})$ is prime (or nearly prime). Note that there are 6 possible group orders for $y^2 = x^3 + B'$ over $\mathbb{F}_{p^2}$ and three of them are never prime as they correspond to group orders of curves defined over $\mathbb{F}_p$.

The isomorphism $\phi : E \to E'$ is given by $\phi(x, y) = (u^2 x, u^3 y)$ and is defined over $\mathbb{F}_{p^{12}}$. The homomorphism $\psi = \phi\pi\phi^{-1}$, where $\pi$ is the $p$-power Frobenius on $E$, is defined over $\mathbb{F}_{p^2}$ and satisfies the characteristic equation

$$\psi^4 - \psi^2 + 1 = 0$$

corresponding to the 12-th cyclotomic polynomial. Hence one obtains a 4-dimensional GLV method for these curves. This leads, once again, to a significant speedup of these curves compared with previous techniques.

Note that $-\psi^2$ satisfies the characteristic equation $x^2 + x + 1$ and so acts as the standard automorphism $(x, y) \mapsto (\zeta_3 x, y)$ on $E$.

## 5 Remarks on Our Implementation

In this section we briefly describe the implementation we used for our experiments. As mentioned in the introduction, we do not claim that our implementation is the best possible. We believe that, for the parameters and implementation platforms considered in this paper, it gives a fair estimate of the speedup obtained by using the GLV method.

The main point of the GLV method is to replace a large point multiplication $[n]P$ by a multiexponentiation $[n_0]P + [n_1]\psi(P)$. There are numerous algorithms for multiexponentiation, all built on a fundamental observation by Straus, and much has been written on the topic. One approach is to use 'interleaving'; this idea seems to have been independently discovered in [15] and [24]. We refer to Section 3.3.3 of [18] for details. Another approach is the joint sparse form (see Solinas [34]). The full version of the paper contains further analysis of multiexponentiation methods (e.g., higher-dimensional joint sparse forms, the Euclidean Montgomery ladder etc).

Two fundamental ideas used to speed up the computation of $[n]P$ on elliptic curves are the use of signed binary expansions (for example, non-adjacent forms, see Definition 3.28 [18] or Definition 9.13 of [3]) and sliding window methods. A very efficient method (as it only uses a few word operations) to compute the NAF of an integer $n$ is to compute $3n$ (using standard integer multiplication), then form the signed expansion $(3n) - n$ and discard the least significant bit. The natural extension of non-adjacent forms to windows is called width-$w$ NAFs (see Section IV.2.5 of [9], Definition 3.32 of [18] or Definition 9.19 of [3]). Instead of using width-$w$ NAFs one can use sliding windows over NAF expansions (see Section IV.2.4 of [9] or Algorithm 3.38 on page 101 of [18]). This is convenient since it is cheaper to compute a NAF than a width-$w$ NAF.

More generally, one can use signed fractional windows [25,26]. Finally, one could consider fractional sliding windows over NAFs. This does not seem to have been considered in the literature and it is an open problem to determine the density in this case. More details of these methods are given in the full version of the paper.

Our implementation uses interleaving with sliding (non-fractional) windows of width $w = 4$ over NAF expansions (we found that using $w = 5$ was slightly slower for our parameters). Hence we must precompute $\{P, [3]P, [5]P, [7]P, [9]P\}$; note that the points $\{\psi(P), [3]\psi(P), [5]\psi(P), [7]\psi(P), [9]\psi(P)\}$ can be obtained on the fly at little cost. This is very similar to Algorithm 3.51 of [18], which uses interleaving over width-$w$ NAFs (the authors of [18] tell us that there is a typo in line 2 of Algorithm 3.5.1: one should replace "3.30" with "3.35"). We do not claim that this is the fastest possible approach, but it requires relatively little precomputation and is very simple to implement. It is possible that one could obtain slightly faster results using fractional windows or other methods.

The next decision is which coordinate system to use for elliptic curve arithmetic. The best choice is probably inverted Edwards or Jacobi quartic [6,7,8] but for legacy reasons our implementation uses Jacobian coordinates. As usual, one prefers to use mixed additions in the main loop as they are faster. However this requires that any precomputed values must be "normalized", that is converted to affine form, before entering the loop. This conversion, if done naively for each precomputed point, would require expensive field inversions, so we use the precomputation strategy of Dahmen, Okeya and Schepers (DOS) [12], as recommended in [8] (there are also recent improvements due to Longa and Miri [23]), which requires only a single inversion.

The full version of the paper gives more details of the implementation, as well as a theoretical estimate of the number of $\mathbb{F}_{p^2}$ operations required for our algorithm.

## 6 Experimental Results

We now give some timing comparisons for the computation of $[n]P$ (and also signature verification) on elliptic curves at the 128-bit security level. Our timings are for the case of quadratic twists as presented in Section 2.1.

### 6.1 The Example Curve

It is natural to use the Mersenne prime $p = 2^{127} - 1$, which is also used in Bernstein's `surface1271` genus 2 implementation [5][1]. This prime supports a very fast modular reduction algorithm.

Since $p \equiv 3 \pmod 4$ we represent $\mathbb{F}_{p^2}$ as $\mathbb{F}_p(\sqrt{-1})$. Note that since $p \neq 5 \bmod 8$ the previously described key generation process is not applicable here. However it can easily be modified to handle this case as well, although the homomorphism requires more multiplications to compute.

Let

$$E : y^2 = x^3 - 3x + 44$$

[1] Note that the Pollard rho algorithm using equivalence classes in this case requires approximately $2^{125}$ group operations, the same as for Bernstein's Curve25519 or Surface1271. Whether this is precisely the same security level as AES-128 is unclear, but since Curve25519 and Surface1271 have been used for benchmarking we feel our choice is justified.

be defined over the field $\mathbb{F}_p$. Then $\#E(\mathbb{F}_p) = p+1-t$ where $t =$ `3204F5AE088C39A7` in hex. By Corollary 1 the quadratic twist $E'$ over $\mathbb{F}_{p^2}$ of $E(\mathbb{F}_{p^2})$ has $\#E'(\mathbb{F}_{p^2}) = (p-1)^2+t^2$, which is a prime we call $r$. The curve was quickly found using a modified version of Schoof's algorithm.

We use $u = 2 + i$ instead of $u = \sqrt{2}$ in Algorithm 1. The homomorphism in this case simplifies to

$$\psi(x, y) = (\omega_x \bar{x}, \omega_y \bar{y})$$

where $\bar{x}$ denotes the Galois conjugate of $x$, and $\omega_x = u/u^p, \omega_y = \sqrt{u^3/u^{3p}}$ as in the proof of Corollary 1. By Lemma 2 we have $\psi(P) = [\lambda]P$ where $\lambda = t^{-1}(p-1) \pmod{r}$.

### 6.2 Comparison Curve

For comparison purposes we consider an elliptic curve $E$ defined over $\mathbb{F}_{p_2}$ where $p_2 = 2^{256} - 189$ is a 256-bit pseudo-Mersenne modulus. This provides approximately the same level of security as the curve in the previous subsection.

The full version of the paper gives a theoretical comparison of the implementations. Table 1 gives operation counts for our test implementation. The notation SSW means sliding windows of window size $w = 5$ over NAFs, GLV+JSF means using joint sparse forms for the multiexponentiation and GLV+INT means interleaving sliding windows of size 4 over NAFs as described in Section 5. In our implementations we averaged the cost over $10^5$ point multiplications.

**Table 1.** Point multiplication operation counts

| | Method | $\mathbb{F}_p$ muls | $\mathbb{F}_p$ adds/subs |
|---|---|---|---|
| $E(\mathbb{F}_{p_2})$, 256-bit $p_2$ | SSW | 2600 | 3775 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | SSW | 6641 | 16997 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | GLV+JSF | 4423 | 10785 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | GLV+INT | 4109 | 10112 |

The results in Table 1 agree with the rough analysis given in the full version of the paper. The table includes the often neglected costs of field additions and subtractions. Note that when implementing $\mathbb{F}_{p^2}$ arithmetic, each multiplication using Karatsuba requires five $\mathbb{F}_p$ additions or subtractions (assuming $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{-1})$), so the number of these operations increases substantially.

Clearly the superiority (or otherwise) of the method depends on the relative cost of 128-bit and 256-bit field multiplications (and additions or subtractions) on the particular platform.

To give a more accurate picture we have implemented both methods on two widely differing platforms, a 1.66GHz 64-bit Intel Core 2, and on an 8-bit 4MHz Atmel Atmega1281 chip (which is a popular choice for wireless sensor network nodes). We present the results in the following two subsections.

### 6.3 8-bit Processor Implementation

Our first implementation is on a small 4MHz 8-bit Atmega1281 processor. Here the base field multiplication times will dominate, so this function was written in optimal loop-unrolled assembly language. We use the MIRACL C library [31], which includes tools for the automatic generation of such code (and which holds the current speed record for this particular processor [32]), and we use the cycle accurate AVR Studio tool to measure the time for a single variable point multiplication.

**Table 2.** Point multiplication timings – 8-bit processor

| Atmel Atmega1281 processor | Method | Time (s) |
|---|---|---|
| $E(\mathbb{F}_{p_2})$, (256-bit $p_2$) | SSW | 5.49 |
| $E(\mathbb{F}_{p^2})$ (127-bit $p$) | SSW | 6.20 |
| $E(\mathbb{F}_{p^2})$, (127-bit $p$) | GLV+JSF | 4.21 |
| $E(\mathbb{F}_{p^2})$, (127-bit $p$) | GLV+INT | 3.87 |

Table 2 show a that our best method for point multiplication takes about 0.70 of the time required for the 256 bit $E(\mathbb{F}_{p_2})$ curve.

Observe that simply switching to an $E(\mathbb{F}_{p^2})$ curve at the same security level does not by itself give any improvement, in fact it is somewhat slower. The theoretical advantage of using Karatsuba in the latter case appears to be outweighed by the extra “fussiness” of the $\mathbb{F}_{p^2}$ implementation; and of course Karatsuba can also be applied to the $\mathbb{F}_p$ case as well if considered appropriate. Looking at the timings, a field multiplication takes 1995 $\mu$s over $\mathbb{F}_{p_2}$ (256-bit), as against 2327 $\mu$s over $\mathbb{F}_{p^2}$ (127-bit $p$), although for a field squaring the situation is reversed, taking 1616 $\mu$s over $\mathbb{F}_{p_2}$ as against only 1529 $\mu$s over $\mathbb{F}_{p^2}$. Field addition and subtraction favours the $\mathbb{F}_{p_2}$ case (124 $\mu$s versus 174 $\mu$s). However using the new homomorphism and applying the GLV method, our new implementation is still clearly superior.

Note that for this processor it is probably more appropriate in practice to use the JSF method for point multiplication, as it is much better suited to a small constrained enviroment, with limited space for online precomputation.

### 6.4 64-Bit Processor Implementation

It has been observed by Avanzi [2], that software implementations over smaller prime fields, where field elements can be stored in just a few CPU registers (as will be the case here), suffer disproportionally when implemented using general purpose multi-precision libraries. This effect would work against us here, as we are using the general purpose MIRACL library [31]. Special purpose libraries like the `mpFq` library [17] which generate field-specific code, and implementations which work hard to squeeze out overheads, such as Bernstein’s implementations [5] are always going to be faster.

In the context of a 64-bit processor, while one might hope that timings would be dominated by the $O(n^2)$ base field multiplication operations, for small values of $n$ the $O(n)$ contribution of the numerous base field additions and subtractions becomes significant, as also observed by Gaudry and Thomé [17]. Observe that on the 64-bit processor a 128-bit field element requires just $n = 2$ (and indeed the description as "multi-precision" should really give way to "double precision"). Therefore it is to be expected that the speed-up we can achieve in this case will be less than might have been hoped.

So is our new method faster? There is really only one satisfactory way to resolve the issue – and that is to identify the fastest known $E(\mathbb{F}_{p_2})$ implementation on a 64-bit processor for the same level of security, and try to improve on it. We understand that the current record is that announced by Gaudry and Thomé at SPEED 2007 [17], using an implementation of Bernstein's `curve25519` [4]. This record is in the setting of an implementation of the elliptic curve Diffie-Hellman method, which requires a single point multiplication to determine the shared secret key.

We point out that the clever implementation and optimizations of `curve25519` are for the sole context of an efficient Diffie-Hellman implementation – ours is general purpose and immediately applicable to a wide range of ECC protocols. In particular the implementation of `curve25519` uses Montgomery's parameterisation of an elliptic curve, is not required to maintain a $y$ coordinate, and hence can achieve compression of the public key at no extra cost (i.e., without the calculation of a square root).

On the other hand we have the use of a particularly nice modulus $2^{127} - 1$, which brings many benefits. For example a base field square root of a quadratic residue $x$ can be calculated as simply $x^{2^{125}}$.

In order to be competitive we wrote a specialised hand-crafted x86-64 assembly language module to handle the base field arithmetic, and integrated this with the MIRACL library. Given that each field element can be stored in just two 64-bit registers, this code is quite short, and did not take long to generate, optimize and test.

To obtain our timings we follow Gaudry and Thomé, and utilise two different methods, one based on actual cycle counts, and a method which uses an operating system timer. There are problems with both methods [17], so here we average the two. In practise the two methods were in close agreement, but not of sufficient accuracy to justify exact numbers – so we round to the nearest 1000 cycles. See

**Table 3.** Point multiplication timings – 64-bit processor

| Intel Core 2 processor | Method | Clock cycles |
|---|---|---|
| $E(\mathbb{F}_{p_2})$, 255-bit $p_2$ | Montgomery [17] | 386,000 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | SSW | 490,000 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | GLV+JSF | 359,000 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | GLV+INT | 326,000 |

Table 3 for our results. As can be seen, our best method takes 0.84 of the time of the Gaudry and Thomé implementation. Note that point decompression, as required by a Diffie-Hellman implementation which wishes to minimise the size of the public key, would require approximately an extra 26,000 clock cycles for our implementation.

It is interesting to observe from Table 3 that a careful implementation over a quadratic extension which does not exploit our homomorphism is substantially slower, taking 490,000 cycles. So again it seems that merely switching to a smaller field size is not by itself advantageous on a 64-bit processor, although some of the difference can be explained by the particularly clever parameterization chosen for `curve25519`. However by using the GLV method we are able to make up this difference, and indeed overtake the previous record.

To ensure a fair comparison, we exploited the very useful `eBats` project [10] (now incorporated into eBACS [11]). Our eBat implements a Diffie-Hellman key exchange algorithm, and can be directly and independently compared with an implementation based on `curve25519`. There are two main functions for a Diffie-Hellman implementation, one which calculates the key pair, and a second which calculates the shared secret. For the key pair calculation we exploit the fact that for our method a multiplication of a fixed point can benefit from extensive offline precomputation, and use a fixed-base comb algorithm (see Section 3.3.2 of [18]), and so this calculation requires only 146,000 cycles. For the shared secret calculation we use the GLV+INT method, plus the cost of a point decompression.

Our latest eBat can be downloaded from:
`ftp://ftp.computing.dcu.ie/pub/crypto/gls1271-3.tar`
Profiling the code reveals that our version (with point compression) spends 49% of its time doing base field multiplications and squarings, 15% of the time doing base field additions and subtractions and nearly 6% of the time is required for the few modular inversions.

### 6.5 ECDSA/Schnorr Signature Verification

Verification of both ECDSA and Schnorr signatures requires the calculation of $[a]P+[b]Q$, where $P$ is fixed. In our setting we must calculate $[a_0]P+[a_1]\psi(P)+[b_0]Q+[b_1]\psi(Q)$ – in other words a 4-dimensional multiexponentiation algorithm is required. The methods of Bernstein [4] and Gaudry-Thomé [17] are based on Montgomery arithmetic and are not appropriate for signature verification.

Again we use an interleaving algorithm, using windows over a NAF expansion. Since $P$ is now fixed, precomputation of multiples of $P$ (and therefore of $\psi(P)$) can be carried out offline, and so a larger window size of 6 can be used for the multiplication of $P$. This requires the precomputation and storage of 42 points. For the online precomputation required on $Q$, we again use sliding windows of size 4 over NAF expansions.

In Table 4 we compare our method with an implementation that does not use the GLV method. The notation GLV+INT means a 4-dimensional multiexponentiation as described above and the notation INT means the 2-dimensional interleaving algorithm which calculates $[a]P+[b]Q$ directly for random $a, b < r$,

**Table 4.** Signature Verification timings – 64-bit processor

| Intel Core 2 processor | Method | $\mathbb{F}_p$ muls | $\mathbb{F}_p$ adds/subs | Clock cycles |
|---|---|---|---|---|
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | GLV+INT | 5174 | 12352 | 425,000 |
| $E(\mathbb{F}_{p^2})$, 127-bit $p$ | INT | 7638 | 19046 | 581,000 |

using size 6 sliding windows over NAFs for the fixed point $P$, and size 5 sliding windows over NAFs for the variable point $Q$.

Antipa et al [1] propose a variant of ECDSA with faster signature verification (note that their method does not apply to Schnorr signatures). The basic method gives essentially the same performance as our method (they transform $[a]P+[b]Q$ to a 4-dimensional multiexponentiation with coefficients $\approx \sqrt{r}$). Their method, as with ours, assumes that $P$ is fixed and that certain precomputation has been done.

The paper [1] also gives a variant where the public key is doubled in size to include $Q$ and $Q_1 = [2^{\lceil \log_2(r)/3 \rceil}]Q$. Their method transforms $[a]P + [b]Q$ to a 6-dimensional multiexponentiation with coefficients of size $\approx r^{1/3}$. In this context (i.e., enlarged public keys) we can improve upon their result. Let $M = 2^{\lceil \log_2(r)/4 \rceil}$ and suppose the public key features $Q$ and $Q_1 = [M]Q$. The GLV idea transforms $[a]P+[b]Q$ to $[a_0]P+[a_1]\psi(P)+[b_0]Q+[b_1]\psi(Q)$ where $a_0, a_1, b_0, b_1 \approx \sqrt{r}$. We now write $a_0 = a_{0,0} + M a_{0,1}$ where $a_{0,0}, a_{0,1} \approx r^{1/4}$ and similarly for $a_1, b_0, b_1$. Hence the computation becomes an 8-dimensional multiexponentiation with coefficients of size $\approx r^{1/4}$. Another advantage of our method is that it applies to Schnorr signatures whereas the method of [1] is only for ECDSA and other variants of ElGamal signatures.

Finally, we mention that the methods in [27] can also be applied in our setting.

## 7 Security Implications

The homomorphism $\psi$ of Theorem 1 (at least, in the case when $\phi$ is an isomorphism) defines equivalence classes of points in $E'(\mathbb{F}_{p^m})$ of size $2m$ by $[P] = \{\pm\psi^i(P) : 0 \leq i < m\}$. By the methods of Gallant-Lambert-Vanstone [14] and Wiener-Zuccherato [35] one can perform the Pollard rho algorithm for the discrete logarithm problem on these equivalence classes. This speeds up the solution of the discrete logarithm problem by a factor of $\sqrt{m}$ compared with general curves. Hence one bit should be added to the key length to compensate for this attack.

A more serious threat comes from the Weil descent philosophy, and in particular the work of Gaudry [16]. Gaudry gives an algorithm for the discrete logarithm problem in $E'(\mathbb{F}_{p^m})$ requiring time $O(p^{2-4/(2m+1)})$ group operations (with bad constants) which, in principle, beats the Pollard methods for $m \geq 3$. The proposal for elliptic curves in the case $m = 2$ is immune to Gaudry's Weil descent attack.

Gaudry's method also applies to abelian varieties: if $A$ is an abelian varitey of dimension $d$ over $\mathbb{F}_{p^m}$ then the algorithm has complexity $O(p^{2-4/(2dm+1)})$. Hence, for Jacobians of genus 2 curves over $\mathbb{F}_{p^2}$ one has an algorithm running in time $O(p^{1.55})$, rather than the Pollard complexity of $O(p^2)$. Gaudry's method is exponential time and so one can secure against it by increasing the field size. For example, to achieve 128-bit security level with genus 2 curves over $\mathbb{F}_{p^2}$ or elliptic curves over $\mathbb{F}_{p^4}$ one should take $p$ to be approximately 80 bits rather than the desired 64 bits (this is a very conservative choice; Gaudry's algorithm requries expensive computations such as Gröbner bases and so one can probably safely work with primes smaller than 80 bits).

## Acknowledgements

We thank Dan Bernstein, Billy Brumley, Jinhui Chao, Pierrick Gaudry, Darrel Hankerson, Alfred Menezes, Yasuyuki Nogami, Fre Vercauteren and the anonymous referees for suggestions and comments.

## References

1. Antipa, A., Brown, D., Gallant, R.P., Lambert, R., Struik, R., Vanstone, S.A.: Accelerated Verification of ECDSA Signatures. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 307–318. Springer, Heidelberg (2006)
2. Avanzi, R.M.: Aspects of Hyperelliptic Curves over Large Prime Fields in Software Implementations. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 148–162. Springer, Heidelberg (2004)
3. Avanzi, R., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Cryptography. Chapman and Hall/CRC (2006)
4. Bernstein, D.J.: Curve25519: New Diffie-Hellman Speed Records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
5. Bernstein, D.J.: Elliptic vs. Hyperelliptic, part 1 ECC, Toronto, Canada (2006), `http://www.cacr.math.uwaterloo.ca/conferences/2006/ecc2006/slides.html`
6. Bernstein, D.J., Lange, T.: Faster Addition and Doubling on Elliptic Curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
7. Bernstein, D.J., Lange, T.: Inverted Edwards Coordinates. In: Boztaş, S., Lu, H.-F. (eds.) AAECC 2007. LNCS, vol. 4851, pp. 20–27. Springer, Heidelberg (2007)
8. Bernstein, D.J., Lange, T.: Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication. In: Finite Fields and Applications: Proceedings of Fq8, Contemporary Mathematics 461, pp. 1–18. American Mathematical Society (2008)
9. Blake, I., Seroussi, G., Smart, N.P. (eds.): Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (1999)
10. eBATS: ECRYPT Benchmarking of Asymmetric Systems, `http://www.ecrypt.eu.org/ebats/`
11. Bernstein, D.J., Lange, T.: eBACS: ECRYPT Benchmarking of Cryptographic Systems (accessed January 9, 2009), `http://bench.cr.yp.to/`

12. Dahmen, E., Okeya, K., Schepers, D.: Affine Precomputation with Sole Inversion in Elliptic Curve Cryptography. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 245–258. Springer, Heidelberg (2007)
13. Galbraith, S.D., Scott, M.: Exponentiation in Pairing-Friendly Groups Using Homomorphisms. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 211–224. Springer, Heidelberg (2008)
14. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Improving the Parallelized Pollard Lambda Search on Anomalous Binary Curves. Math. Comp. 69, 1699–1705 (2000)
15. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)
16. Gaudry, P.: Index Calculus for Abelian Varieties of Small Dimension and the Elliptic Curve Discrete Logarithm Problem. J. Symbolic Comput. (to appear)
17. Gaudry, P., Thome, E.: The mpFq Library and Implementing Curve-Based Key Exchanges. In: SPEED workshop presentation, Amsterdam (June 2007)
18. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to elliptic curve cryptography. Springer, Heidelberg (2004)
19. Hankerson, D., Karabina, K., Menezes, A.J.: Analyzing the Galbraith-Lin-Scott Point Multiplication Method for Elliptic Curves over Binary Fields, eprint 2008/334
20. Iijima, T., Matsuo, K., Chao, J., Tsujii, S.: Costruction of Frobenius Maps of Twist Elliptic Curves and its Application to Elliptic Scalar Multiplication. In: SCIS 2002, IEICE Japan, pp. 699–702 (January 2002)
21. Kim, D., Lim, S.: Integer Decomposition for Fast Scalar Multiplication on Elliptic Curves. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 13–20. Springer, Heidelberg (2003)
22. Kozaki, S., Matsuo, K., Shimbara, Y.: Skew-Frobenius Maps on Hyperelliptic Curves, IEICE Trans. E91-A(7), 1839–1843 (2008)
23. Longa, P., Miri, A.: New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
24. Möller, B.: Algorithms for Multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
25. Möller, B.: Improved Techniques for Fast Exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
26. Möller, B.: Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation. In: Park, C.-S., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 137–153. Springer, Heidelberg (2005)
27. Möller, B., Rupp, A.: Faster Multi-exponentiation through Caching: Accelerating (EC)DSA Signature Verification. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 39–56. Springer, Heidelberg (2008)
28. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Math. Comp. 47, 243–264 (1987)
29. Nogami, Y., Morikawa, Y.: Fast Generation of Elliptic Curves with Prime Order over Extension Field of Even Extension Degree. In: Proceedings 2003 IEEE International Symposium on Information Theory, p. 18 (2003)
30. Park, Y.-H., Jeong, S., Kim, C.-H., Lim, J.-I.: An Alternate Decomposition of an Integer for Faster Point Multiplication on Certain Elliptic Curves. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 323–334. Springer, Heidelberg (2002)

31. Scott, M.: MIRACL – Multiprecision Integer and Rational Arithmetic C/C++ Library (2008), http://ftp.computing.dcu.ie/pub/crypto/miracl.zip
32. Scott, M., Szczechowiak, P.: Optimizing Multiprecision Multiplication for Public Key Cryptography (2007), http://eprint.iacr.org/2007/299
33. Sica, F., Ciet, M., Quisquater, J.-J.: Analysis of the Gallant-Lambert-Vanstone Method based on Efficient Endomorphisms: Elliptic and Hyperelliptic Curves. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 21–36. Springer, Heidelberg (2003)
34. Solinas, J.A.: Low-Weight Binary Representations for Pairs of Integers, Technical Report CORR 2001–41, CACR (2001)
35. Wiener, M., Zuccherato, R.J.: Faster Attacks on Elliptic Curve Cryptosystems. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 190–200. Springer, Heidelberg (1999)