ANALYZING THE GALBRAITH-LIN-SCOTT POINT MULTIPLICATION METHOD FOR ELLIPTIC CURVES OVER BINARY FIELDS

DARREL HANKERSON, KORAY KARABINA, AND ALFRED MENEZES

Abstract. Galbraith, Lin and Scott recently constructed efficiently-computable endomorphisms for a large family of elliptic curves defined over $\mathbb{F}_{q^2}$ and showed, in the case where q is prime, that the Gallant-Lambert-Vanstone point multiplication method for these curves is significantly faster than point multiplication for general elliptic curves over prime fields. In this paper, we investigate the potential benefits of using Galbraith-Lin-Scott elliptic curves in the case where q is a power of 2. The analysis differs from the q prime case because of several factors, including the availability of the point halving strategy for elliptic curves over binary fields. Our analysis and implementations show that Galbraith-Lin-Scott offers significant acceleration for curves over binary fields, in both doubling- and halving-based approaches. Experimentally, the acceleration surpasses that reported for prime fields (for the platform in common), a somewhat counterintuitive result given the relative costs of point addition and doubling in each case.

1. Introduction

Let E be a Koblitz elliptic curve, i.e., the elliptic curve $Y^2 + XY = X^3 + 1$ or $Y^2 + XY = X^3 + X^2 + 1$ defined over $\mathbb{F}_2$. Koblitz [24] (see also [36]) showed how point multiplication in $E(\mathbb{F}_{2^n})$ can be accelerated by exploiting the Frobenius endomorphism $\pi : (x, y) \mapsto (x^2, y^2)$. Namely, if $P \in E(\mathbb{F}_{2^n})$ and $k \in \mathbb{Z}$, then one first writes $k = \sum k_i \pi^i$ for $k_i \in \{0, \pm 1\}$, and thereafter computes $kP = \sum k_i \pi^i(P)$. This yields a point multiplication algorithm that is faster than the traditional point multiplication methods because $\pi(P)$ can be computed much faster than $2P$.

Gallant, Lambert and Vanstone (GLV) [11] showed how efficiently-computable endomorphisms can also be used to accelerate point multiplication on certain ordinary elliptic curves defined over finite fields $\mathbb{F}_q$ of characteristic greater than 3 (cf. §2.3). The GLV technique is usually applied to elliptic curves $Y^2 = X^3 + aX$ defined over $\mathbb{F}_q$ where $q \equiv 1 \pmod 4$, or elliptic curves $Y^2 = X^3 + b$ defined over $\mathbb{F}_q$ where $q \equiv 1 \pmod 3$. The first curve has an efficiently-computable endomorphism $\psi : (x, y) \mapsto (-x, iy)$ where $i \in \mathbb{F}_q$ has multiplicative order 4, and the second curve has an efficiently-computable endomorphism $\psi : (x, y) \mapsto (\beta x, y)$ where $\beta \in \mathbb{F}_q$ has multiplicative order 3. These curves are special in that they have complex multiplication by orders in $\mathbb{Q}(i)$ and $\mathbb{Q}(\sqrt{-3})$, respectively. They are also rare in the sense that there are, respectively, only 4 and 6 isomorphism classes of these kinds of curves over a field $\mathbb{F}_q$, whereas the total number of isomorphism classes of elliptic curves over $\mathbb{F}_q$ is approximately $2q$.

Recently, Galbraith, Lin and Scott (GLS) [10] constructed efficiently-computable endomorphisms for a large family of elliptic curves defined over $\mathbb{F}_{q^2}$ (these endomorphisms had been discovered earlier by Iijima, Matsuo, Chao and Tsujii [18]) and showed that the GLV technique can be used to significantly speed point multiplication on these curves in the case where q is a prime number and the wordlength of the software platform is small. The GLS elliptic curves are quite special in that their j-invariants belong to $\mathbb{F}_q$. Nevertheless, since there are approximately q isomorphism classes

Date: August 1, 2008.

of such elliptic curves (out of the roughly $2q^2$ elliptic curves defined over $\mathbb{F}_{q^2}$), the class of elliptic curves for which the GLV technique is effective has been significantly enlarged.

In this paper, we investigate the potential advantages of GLS elliptic curves defined over binary fields $\mathbb{F}_{2^{2\ell}}$. In particular, we wish to determine whether GLV point multiplication is faster than the traditional double-and-add point multiplication methods for these curves, and similarly for halve-and-add methods [23, 34], for the GLS curves themselves and also against general elliptic curves over $\mathbb{F}_{2^n}$ for prime $n \approx 2\ell$. An affirmative answer would mean that there is a large class of elliptic curves over binary fields for which GLV point multiplication is effective. Our analysis and experiments show that the GLS curves offer a significant performance advantage using the GLV technique, under both doubling-based and halving-based point multiplication methods. In comparison to the prime field case examined in [10], the experimental results are somewhat counterintuitive in that acceleration against general elliptic curves is larger for the binary case (on "workstation class" processors) even though the cost of point addition relative to doubling is higher (and it is the number of point doubles or halvings that are reduced by GLV). While binary fields have been less attractive on many processors due to availability of fast integer multipliers, the interest is likely to increase with the characteristic 2 multiplier on 64-bit operands announced by Intel for upcoming processors, since performance profiles will change dramatically [13, 14].

The remainder of this paper is organized as follows. In §2, we review the double-and-add, halve-and-add, and GLV methods for elliptic curve point multiplication. An explicit formulation of the efficiently-computable endomorphism for elliptic curves over binary fields $\mathbb{F}_{q^2}$ is developed in §3. Resistance of GLS curves to Weil descent attacks on the elliptic curve discrete logarithm problem is studied in §4. The point multiplication methods are compared in §5, and experimental results are presented in §6. Finally, we make some concluding remarks in §7.

2. Point multiplication methods

Consider the ordinary elliptic curve

$$E/\mathbb{F}_{2^n} \ : \ Y^2 + XY = X^3 + aX^2 + b$$

defined over a binary field $\mathbb{F}_{2^n}$, and suppose that $\#E(\mathbb{F}_{2^n}) = hr$ where r is prime and h is small (so $r \approx 2^n$). Let $\mathcal{S}$ denote the unique order-r subgroup of $E(\mathbb{F}_{2^n})$ This section reviews the double-and-add, halve-and-add, and GLV methods for computing kP, where $P \in \mathcal{S}$ and $k \in [0, r-1]$.

We denote by m the cost of performing a multiplication in $\mathbb{F}_{2^n}$, and by A and D, respectively, the cost of performing addition and doubling in $E(\mathbb{F}_{2^n})$. We have $D \approx 4m$ when López-Dahab projective coordinates [27] are used to represent points, and $A \approx 8m$ when mixed affine-projective coordinates are employed.

2.1. **Double-and-add.** Let $w \geq 2$ be a positive integer. The *width-w NAF* of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where each nonzero coefficient k_i is odd, $|k_i| < 2^{w-1}$, $k_{l-1} \neq 0$, and at most one of any w consecutive digits is nonzero. The length l of the width-w NAF is at most one more than the length of the binary representation of k, and the average density of nonzero digits among all width-w NAFs of length l is approximately $1/(w+1)$ [36, 33].

To compute kP, one first determines the width-w NAF of k, computes $P_i = iP$ for $i \in \{1, 3, 5, \ldots, 2^{w-1} - 1\}$, and initializes an accumulator to ∞. The digits of the width-w NAF are then examined from left to right. For each digit k_i, the accumulator is doubled and $\pm P_{k_i}$ is added to it (if $k_i \neq 0$). The expected cost of computing kP can be seen to be approximately

$$\left[1D + (2^{w-2} - 1)A\right] + \left[\frac{n}{w+1}A + nD\right]. \tag{1}$$

2.2. **Halve-and-add.** Halve-and-add was proposed independently by Knudsen [23] and Schroeppel [34]. The idea is to replace almost all point doublings in double-and-add methods with a potentially faster operation called point halving. To simplify the exposition we assume that n is odd and $\mathrm{Tr}(a) = 1$, where $\mathrm{Tr} : x \mapsto \sum_{i=0}^{n-1} x^{2^i}$ denotes the trace function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$.

Point halving is the following operation: given $Q = (u, v) \in \mathcal{S}$, compute the unique point $R = (x, y) \in \mathcal{S}$ such that $Q = 2R$. Recall that the (affine) coordinates of Q can be obtained from those of R via the doubling formulae: $\lambda = x + y/x$, $u = \lambda^2 + \lambda + a$, $v = x^2 + u(\lambda + 1)$. To recover R from Q, one first finds a solution $\hat{\lambda}$ to the quadratic equation $\hat{\lambda}^2 + \hat{\lambda} = u + a$. Then one has $\lambda = \hat{\lambda} + \mathrm{Tr}(t)$ where $t = v + u\hat{\lambda}$ [23]. One can then compute $x = \sqrt{t+u}$ or $x = \sqrt{t}$ if $\mathrm{Tr}(t) = 0$ or 1, and $y = \lambda x + x^2$. In the halve-and-add algorithm described in the next paragraph, repeated halving of Q is required. To accelerate halving, one uses the λ-representation (u, λ_Q) of Q, where $\lambda_Q = u + v/u$. The revised formula for t is $t = u(u + \lambda_Q + \hat{\lambda})$; note that y does not have to be computed (thus saving a multiplication).

To compute kP, one first finds the w-NAF representation $\sum_{i=0}^{l} k_i' 2^i$ of $2^{l-1}k \bmod n$; here $l \approx n$ is the bitlength of k. Then $k \equiv \sum_{i=0}^{l-1} k_{l-1-i}'/2^i + 2k_l' \pmod{n}$, whence $kP = \sum_{i=0}^{l-1} k_{l-1-i}'(\frac{1}{2^i}P) + 2k_l'P$. The halve-and-add algorithm for computing kP is analogous to the double-and-add method, and has an expected cost of approximately

$$\left[1D + (2^{w-2} - 1)A\right] + \left[\frac{n}{w+1}\tilde{A} + nH\right], \tag{2}$$

where $\tilde{A} = A + m$ is the cost of a point addition when one of the inputs is in λ-representation, and H is the cost of a halving. Square roots in $\mathbb{F}_{2^n}$ can be computed very efficiently, especially when the reduction polynomial selected to represent $\mathbb{F}_{2^n}$ is judiciously chosen [2]. A solution to the quadratic equation $\hat{\lambda}^2 + \hat{\lambda} = c$ can be obtained using the formula $\hat{\lambda} = \sum_{i=0}^{(n-1)/2} c^{2^{2i}}$; the cost of evaluating this expression can be as low as half the cost of a field multiplication if there is sufficient storage for some precomputed values (see [7]). It follows that the point halving cost is $H \approx 2m$. The speed advantage of halve-and-add over double-and-add can now be appreciated by comparing (1) and (2).

2.3. **GLV method.** Suppose that ψ is a (non-trivial) efficiently-computable endomorphism of E defined over $\mathbb{F}_{2^n}$. Let $\lambda \in [2, r-2]$ be the integer such that $\psi(Q) = \lambda Q$ for all $Q \in \mathcal{S}$. Then 'efficiently-computable' means that $\psi(Q)$ can be computed much more efficiently than computing λQ by standard point multiplication techniques. The GLV strategy [11] for computing kP is to first use the extended Euclidean algorithm to write $k = k_1 + k_2\lambda \bmod r$, where $k_1, k_2 \approx \sqrt{r}$, and then compute $kP = k_1P + k_2\psi(P)$ using simultaneous multiple point multiplication (also known as 'Shamir's trick' – see Algorithm 14.88 in [31]) or interleaving [11, 32]. Since the bitlengths of k_1 and k_2 are half that of k, half of the point doublings are eliminated. If interleaving is used, and if k_1 and k_2 are represented in width-w NAFs, then the expected cost of computing kP is approximately

$$2\left[1D + (2^{w-2} - 1)A\right] + \left[\frac{n}{w+1}A + \frac{n}{2}D\right]. \tag{3}$$

3. The endomorphism

3.1. **Weierstrass form.** Let $q = 2^\ell$ and let

$$E/\mathbb{F}_q \ : \ Y^2 + XY = X^3 + aX^2 + b$$

be an ordinary elliptic curve defined over $\mathbb{F}_q$. Recall that the elliptic curve

$$\tilde{E}/\mathbb{F}_q \ : \ Y^2 + XY = X^3 + \tilde{a}X^2 + \tilde{b}$$

is isomorphic to E over $\mathbb{F}_q$ if and only if $\mathrm{Tr}(\tilde{a}) = \mathrm{Tr}(a)$ and $\tilde{b} = b$, where $\mathrm{Tr} : x \mapsto \sum_{i=0}^{\ell-1} x^{2^i}$. If E and $\tilde{E}$ are isomorphic over $\mathbb{F}_q$ then an isomorphism $E \to \tilde{E}$ is given by $(x, y) \mapsto (x, y + \rho x)$ where $\rho \in \mathbb{F}_q$ satisfies $\rho^2 + \rho = a + \tilde{a}$.

Now, suppose that $\#E(\mathbb{F}_q) = q + 1 - t$. Then $\#E(\mathbb{F}_{q^2}) = (q + 1)^2 - t^2$. Let $\mathrm{Tr}' : x \mapsto \sum_{i=0}^{2\ell-1} x^{2^i}$ denote the trace function from $\mathbb{F}_{q^2}$ to $\mathbb{F}_2$, and let $a' \in \mathbb{F}_{q^2}$ be an element with $\mathrm{Tr}'(a') = 1$. Since $\mathrm{Tr}'(a) = 0$, the elliptic curve

$$E'/\mathbb{F}_{q^2} \ : \ Y^2 + XY = X^3 + a'X^2 + b$$

is the *quadratic twist of* E *over* $\mathbb{F}_{q^2}$ and $\#E'(\mathbb{F}_{q^2}) = 2q^2 + 2 - \#E(\mathbb{F}_{q^2}) = (q - 1)^2 + t^2$.

The elliptic curves E and E' are isomorphic over $\mathbb{F}_{q^4}$, with the isomorphism given by

$$\phi : E \to E', \qquad (x, y) \mapsto (x, y + sx),$$

where $s \in \mathbb{F}_{q^4} \setminus \mathbb{F}_{q^2}$ satisfies $s^2 + s = a + a'$. The map ϕ is an involution, so the inverse isomorphism is $\phi^{-1} = \phi$. Define the Frobenius map $\pi : E \to E$ by $(x, y) \mapsto (x^q, y^q)$, and define

$$\psi : E' \to E' \quad \text{by} \quad \psi = \phi\pi\phi^{-1}.$$

More explicitly, we have

$$\psi : (x, y) \mapsto (x^q, y^q + s^q x^q + s x^q).$$

Since

$$s^q + s = \sum_{i=1}^{\ell} (s^{2^i} + s^{2^{i-1}}) = \sum_{i=1}^{\ell} (s^2 + s)^{2^{i-1}} = \sum_{i=1}^{\ell} (a + a')^{2^{i-1}} \in \mathbb{F}_{q^2},$$

the endomorphism ψ is defined over $\mathbb{F}_{q^2}$. Moreover, it is efficiently computable, its cost being roughly equal to a single $\mathbb{F}_{q^2}$ multiplication.

Now, suppose that $\#E'(\mathbb{F}_{q^2}) = hr$ where r is prime and h is small. Let $\mathcal{S}$ denote the unique order-r subgroup of $E'(\mathbb{F}_{q^2})$. If $Q = (x, y) \in \mathcal{S}$, then

$$\psi^2(Q) = \phi\pi^2\phi^{-1}(Q) = (x, y + s^{q^2} x + sx) = (x, y + x) = -Q,$$

since $s^{q^2} + s = \mathrm{Tr}'(a + a') = 1$. Hence $\psi^2 = -1$. Thus, there is an integer λ satisfying $\lambda^2 + 1 \equiv 0 \pmod{r}$ such that $\psi(Q) = \lambda Q$ for all $Q \in \mathcal{S}$.

3.2. **Edwards form.** Bernstein, Lange and Farashahi [4] showed that all elliptic curves over binary fields can be written in Edwards form, which has the advantage of efficient and complete addition formulas. The following shows that the elliptic curve $E'/\mathbb{F}_{q^2}$ and the efficiently-computable endomorphism ψ can be transformed to Edwards form.

Select $d_1 \in \mathbb{F}_{q^2}$ so that $\mathrm{Tr}'(d_1) = \mathrm{Tr}'(a') + 1$ and $\mathrm{Tr}'(\sqrt{b}/d_1^2) = 1$; Theorem 4.3 of [4] shows that such d_1 can be easily found. Let $d_2 = d_1^2 + d_1 + \sqrt{b}/d_1^2$, and define

$$E_2'/\mathbb{F}_{q^2} : Y^2 + XY = X^3 + (d_1^2 + d_2)X^2 + b.$$

Then, since $\mathrm{Tr}'(d_1^2 + d_2) = \mathrm{Tr}'(a')$, we have $E' \cong E_2'$ over $\mathbb{F}_{q^2}$ with isomorphism $\gamma : E' \to E_2'$ defined by $(x, y) \mapsto (x, y + \rho x)$ where $\rho^2 + \rho = a' + d_1^2 + d_2$. Now, according to [4], E_2' is isomorphic over $\mathbb{F}_{q^2}$ to the complete Edwards binary elliptic curve

$$E_e'/\mathbb{F}_{q^2} : d_1(X + Y) + d_2(X^2 + Y^2) = XY + XY(X + Y) + X^2Y^2$$

with isomorphism $\xi : E_2' \to E_e'$ given by

$$(x,y) \mapsto \left(\frac{d_1(x+d_1^2+d_1+d_2)}{x+y+(d_1^2+d_1)(d_1^2+d_1+d_2)} \,,\, \frac{d_1(x+d_1^2+d_1+d_2)}{y+(d_1^2+d_1)(d_1^2+d_1+d_2)} \right)$$

and inverse isomorphism $\xi^{-1} : E_e' \to E_2'$ given by

$$(x,y) \mapsto \left(\frac{d_1(d_1^2+d_1+d_2)(x+y)}{xy+d_1(x+y)} \,,\, d_1(d_1^2+d_1+d_2) \left(\frac{x}{xy+d_1(x+y)} + d_1 + 1 \right) \right).$$

Hence $\psi_e = \xi\gamma\psi\gamma^{-1}\xi^{-1}$ is an efficiently-computable endomorphism on E_e' satisfying $\psi_e(Q) = \lambda Q$ for all order-r points $Q \in E_e'(\mathbb{F}_{q^2})$, and so GLV point multiplication can be effectively carried out in $E_e'(\mathbb{F}_{q^2})$.

4. Security

The Gaudry-Hess-Smart Weil descent attack [12], and its generalization by Hess [17], has been shown to be effective for solving the discrete logarithm problem (DLP) in some elliptic curves over characteristic two finite fields of composite extension degrees. In this section we study the possible effectiveness of Weil descent attacks on GLS elliptic curves over $\mathbb{F}_{2^{2\ell}}$ where ℓ is prime. That is, we examine whether the attacks can be used to solve the DLP faster than it would take Pollard's rho method which has running time approximately 2^ℓ (for the hardest instances).

We begin by introducing some notation. Let $n \in \{2, \ell, 2\ell\}$, and let $q = 2^{2\ell/n}$. Let $K = \mathbb{F}_{2^{2\ell}}$ and $k = \mathbb{F}_q$. Let $\sigma : K \to K$ denote the Frobenius automorphism defined by $\alpha \mapsto \alpha^q$. If $f = \sum_{i=0}^d c_i x^i$ is a polynomial in $\mathbb{F}_2[x]$ and $\gamma \in K$, then we define $f(\sigma)(\gamma) = \sum_{i=0}^d c_i \gamma^{q^i}$. For $\gamma \in K$, we denote by Ord_γ the unique polynomial $f \in \mathbb{F}_2[x]$ of least degree satisfying $f(\sigma)(\gamma) = 0$. It is easy to see that Ord_γ is a divisor of $x^n + 1$, and that $\mathrm{Ord}_\gamma = \mathrm{Ord}_{\gamma^2}$.

Let $E : Y^2 + XY = X^3 + aX^2 + b$ be an elliptic curve defined over $\mathbb{F}_{2^{2\ell}}$. In the following, we are really only interested in those curves for which $\#E(\mathbb{F}_{2^{2\ell}}) = hr$ where r is prime and h is small, whence $r \approx 2^{2\ell}$; these are the curves of interest in cryptographic applications. Since $\mathrm{Tr}'(a) = 1$ for all GLS curves, we fix a to be some element satisfying $\mathrm{Tr}'(a) = 1$ and sometimes denote E by E_b. Note that $\#E(\mathbb{F}_{2^{2\ell}}) \equiv 2 \pmod 4$.

In the generalized Gaudry-Hess-Smart (gGHS) attack [17], one first writes $b = (\gamma_1\gamma_2)^2$, where $\gamma_1, \gamma_2 \in K$ and either $\mathrm{Tr}_{K/k}(\gamma_1) \neq 0$ or $\mathrm{Tr}_{K/k}(\gamma_2) \neq 0$. Let $s_1 = \deg(\mathrm{Ord}_{\gamma_1})$, $s_2 = \deg(\mathrm{Ord}_{\gamma_2})$, and $t = \deg(\mathrm{lcm}(\mathrm{Ord}_{\gamma_1}, \mathrm{Ord}_{\gamma_2}, x+1))$. Then the gGHS attack reduces the DLP in $E(K)$ to the DLP in the jacobian $J_C(k)$ of a curve C of genus $g = 2^t - 2^{t-s_1} - 2^{t-s_2} + 1$ defined over k. The curve C is hyperelliptic if and only if $\gamma_1 \in k$ or $\gamma_2 \in k$. Since $\#J_C(k) \approx q^g$, a necessary condition for $J_C(k)$ to contain a subgroup of order r is $g \geq n$. One can then hope, at least if g is not too large, that the known discrete log algorithms for higher-genus curves could be used to solve the DLP in $J_C(k)$ in time less than 2^ℓ.

Even if the gGHS attack turns out to be ineffective for a particular GLS curve E', it may be effective for an isogenous curve E defined over $\mathbb{F}_{2^{2\ell}}$.[1] In that case, it is generally feasible to map the DLP in $E'(\mathbb{F}_{2^{2\ell}})$ to the DLP in $E(\mathbb{F}_{2^{2\ell}})$ by using a chain of low degree isogenies (see [8] and [19]). Thus, for a chosen GLS curve E', it is important to verify that the gGHS attack is ineffective for all elliptic curves isogenous to E'.

We consider separately the cases $n = 2$, $n = \ell$ and $n = 2\ell$.

[1] Two elliptic curves E_1 and E_2 defined over a finite field K are *isogenous* over K is $\#E_1(K) = \#E_2(K)$. The equivalence classes of elliptic curves with respect to isogeny are called *isogeny classes*.

4.1. **Case 1: $n = 2$ and $q = 2^\ell$.** We have $x^n + 1 = (x + 1)^2$. It follows that we must either take $\mathrm{Ord}_{\gamma_1} = (x + 1)^2$ and $\mathrm{Ord}_{\gamma_2} = (x + 1)^2$ (in which case the gGHS attack produces a genus-3 non-hyperelliptic curve C_1 over $\mathbb{F}_{2^\ell}$), or $\mathrm{Ord}_{\gamma_1} = (x + 1)^2$ and $\mathrm{Ord}_{\gamma_2} = (x + 1)$ (in which case the gGHS attack produces a genus-2 hyperelliptic curve C_2 over $\mathbb{F}_{2^\ell}$). Since the fastest algorithm known for solving the DLP in $J_{C_1}(\mathbb{F}_{2^\ell})$ has running time approximately 2^ℓ [5], and since the fastest algorithm known for the DLP in $J_{C_2}(\mathbb{F}_{2^\ell})$ is Pollard's rho method, the gGHS attack with $n = 2$ is ineffective.

4.2. **Case 2: $n = \ell$ and $q = 2^2$.** Let d denote the multiplicative order of 2 modulo n. We have $x^n+1 = (x+1)f_1f_2\cdots f_s$, where the f_i are pairwise distinct irreducible polynomials of degree d and $s = (n-1)/d$ [30, Lemma 7]. One can check that the selection $\mathrm{Ord}_{\gamma_1} = f_i$ and $\mathrm{Ord}_{\gamma_2} = x+1$ yields the smallest useful genus, namely $g = 2^d - 1$. Now, for each prime $\ell \in [80, 256]$ with $\ell \notin \{89, 127\}$, we have $d \geq 15$. For these ℓ, the gGHS attack produces a curve C of genus at least $2^{15} - 1 = 32767$; the orders of the jacobians $J_C(\mathbb{F}_{2^2})$ have bitlength at least 65534. Since the hardest instances of the DLP in elliptic curves over 512-bit fields is roughly equally as difficult as factoring 15360-bit integers, and since the latter problem is certainly no harder than the DLP in 65534-bit jacobians, it follows that the gGHS attack is ineffective for all $\ell \notin \{89, 127\}$.

For $\ell = 89$, we have $d = 11$. Hence the gGHS attack produces curves C of genus at least $2^{11} - 1 = 2047$. The bitlength of $\#J_C(\mathbb{F}_{2^2})$ is at least 4094. Since the hardest instances of the DLP in elliptic curve over 178-bit fields is certainly no more difficult than factoring 1500-bit integers, the gGHS attack is ineffective for $\ell = 89$.

For $\ell = 127$, we have $d = 7$ and $s = 18$. For some elliptic curves over $\mathbb{F}_{2^{2\ell}}$, the gGHS attack produces a genus-127 hyperelliptic curve C over $\mathbb{F}_{2^2}$. According to the estimates in [29], the Enge-Gaudry index-calculus algorithm [6] can solve the DLP in $J_C(\mathbb{F}_{2^2})$ in time approximately 2^{52}, which is much faster than the 2^{127} time it would take using Pollard's rho method. Thus the gGHS attack is indeed effective for some elliptic curves when $\ell = 127$ and $n = 127$. The vulnerable b's, and upper bounds on their numbers, are listed in Table 1.[2] In total, there are at most 2^{36} vulnerable b's. Note, however, that if b is vulnerable then so is b^2. Moreover, the elliptic curves E_b and E_{b^2} are isogenous over $\mathbb{F}_{2^{254}}$. Hence the number of isogeny classes that contain at least one vulnerable curve is at most $2^{36}/254 \approx 2^{28}$. Now, there are $\approx 2^{127}$ isogeny classes of ordinary elliptic curves over $\mathbb{F}_{2^{254}}$ that have order congruent to 2 modulo 4, of which roughly $\sqrt{2^{127}} = 2^{63.5}$ classes contain a GLS curve. We assume that the probability of a curve E having a vulnerable curve in its isogeny class is independent of whether E is a GLS curve — this is a plausible assumption because there is no reason to believe that the b's in Table 1 are more likely to be in $\mathbb{F}_{2^{127}}$, and because the vulnerability of a curve does not seem to depend on its order. Under this assumption, the probability that a

Table 1. List of vulnerable $b = (\gamma_1\gamma_2)^2$ for the case $\ell = n = 127$, $q = 2^2$.

Ord_{γ_1}	Ord_{γ_2}	s_1	s_2	t	g	Upper bound on number of b's
$(x+1)f_i$	$(x+1)f_i$	8	8	8	255	$s(q^7-1)^2(q-1)^2/2 \approx 2^{35}$
$(x+1)f_i$	f_i	8	7	8	254	$s(q^7-1)(q-1)(q^7-1) \approx 2^{34}$
$(x+1)f_i$	$x+1$	8	1	8	128	$s(q^7-1)(q-1)(q-1) \approx 2^{22}$
f_i	f_i	7	7	8	253	$s(q^7-1)^2/2 \approx 2^{32}$
f_i	$x+1$	7	1	8	127	$s(q^7-1)(q-1) \approx 2^{20}$

[2] For any factor $f \in \mathbb{F}_2[x]$ of $x^n - 1$, the number of $\gamma \in \mathbb{F}_{2^{2\ell}}$ satisfying $\mathrm{Ord}_\gamma = f$ can be easily computed using Theorem 5 of [30].

randomly selected GLS curve has a vulnerable curve in its isogeny class is at most $2^{28}/2^{127} \approx 1/2^{99}$, which is certainly negligible.

For the case $\ell = 127$, an explicit check that a given GLS curve E' does not succumb to the gGHS attack would require enumerating all the (at most) 2^{28} b's in Table 1 (excluding the squares of a selected b), and then checking that $\#E_b(\mathbb{F}_{2^{254}}) \neq \#E'(\mathbb{F}_{2^{254}})$ for each b. The Magma package, running on a 1 GHz Sun V440, can compute the order of a randomly selected elliptic curve over $\mathbb{F}_{2^{254}}$ is about 0.42 seconds. Thus, performing the explicit check would require about 1305 days of CPU time, a feasible task since it can be easily parallelized.

4.3. Case 3: $n = 2\ell$ and $q = 2$.

Let d denote the multiplicative order of 2 modulo n. We have $x^n + 1 = (x+1)^2 f_1^2 f_2^2 \cdots f_s^2$, where the f_i are pairwise distinct irreducible polynomials of degree d and $s = (\ell - 1)/d$. One can check that the gGHS attack produces a curve $C/\mathbb{F}_2$ of genus at least 2^{d+1}. If $\ell \in [80, 256]$ is a prime with $\ell \notin \{89, 127\}$ then $d \geq 15$, while if $\ell = 89$ then $d = 11$. Thus, as in the previous case of $n = \ell$, the gGHS attack is ineffective for all $\ell \neq 127$.

We next consider the case $\ell = 127$ for which $d = 7$ and $s = 18$. For some elliptic curves over $\mathbb{F}_{2^{254}}$, the gGHS attack produces a genus-256 hyperelliptic curve C over $\mathbb{F}_2$. Since the DLP in $J_C(\mathbb{F}_2)$ can be solved much faster than the 2^{127} time it would take using Pollard's rho method, the gGHS attack is indeed effective for some elliptic curves when $\ell = 127$ and $n = 254$. The vulnerable b's, and upper bounds on their numbers, are listed in Table 2. In total, there are at most 2^{21} vulnerable b's. As in the $n = 127$ case, the number of isogeny classes that contain at least one vulnerable curve is at most $2^{21}/254 \approx 2^{13}$. Hence the probability that a randomly selected GLS curve has a vulnerable curve in its isogeny class is at most $2^{13}/2^{127} \approx 1/2^{114}$, which is certainly negligible. Furthermore, one can efficiently verify that there is no vulnerable curve isogenous to a given GLS curve.

TABLE 2. List of vulnerable $b = (\gamma_1\gamma_2)^2$ for the case $\ell = 127$, $n = 254$, $q = 2$.

Ord_{γ_1}	Ord_{γ_2}	s_1	s_2	t	g	Upper bound on number of b's
$(x+1)^2 f_i$	$(x+1)^2 f_i$	9	9	9	511	$18 \cdot 2^{15}$
$(x+1)^2 f_i$	$(x+1) f_i$	9	8	9	510	$18 \cdot 2^{15}$
$(x+1)^2 f_i$	f_i	9	7	9	508	$18 \cdot 2^{15}$
$(x+1)^2 f_i$	$(x+1)^2$	9	2	9	384	$18 \cdot 2^{9}$
$(x+1)^2 f_i$	$x+1$	9	1	9	256	$18 \cdot 2^{8}$
$(x+1) f_i$	$(x+1)^2$	8	2	9	383	$18 \cdot 2^{8}$
f_i	$(x+1)^2$	7	2	9	381	$18 \cdot 2^{8}$

4.4. Summary.

We have shown that GLS curves over $\mathbb{F}_{2^{2\ell}}$ with ℓ prime, $\ell \in [80, 256]$, and $\ell \neq 127$, are not vulnerable to the generalized GHS Weil descent attack. Moreover, the probability that a randomly selected GLS curve over $\mathbb{F}_{2^{254}}$ is vulnerable (or isogenous to a vulnerable curve) is negligibly small, and in any case there is an efficient check that can be performed to rule out this possibility. Hence, GLS curves over $\mathbb{F}_{2^{2\ell}}$ with ℓ prime can be easily selected so that the fastest attack known on the DLP is Pollard's rho method.

5. Comparisons

We examine point multiplication on GLS curves $E'/\mathbb{F}_{2^{2\ell}} : Y^2 + XY = X^3 + a'X^2 + b$ and random curves $E/\mathbb{F}_{2^n} : Y^2 + XY = X^3 + aX^2 + b$ (where $n \approx 2\ell$). For the case of random curves, we will consider the common scenario where n is prime, and where the curve coefficient b is chosen at random subject to the requirement that $\#E(\mathbb{F}_{2^n}) = hr$ where h is 2 or 4 (depending on $\text{Tr}(a)$) and r is prime; without loss of generality we may assume that $a \in \{0, 1\}$. Table 3 summarizes point multiplication costs when parameters have been chosen compatible with the corresponding method, where A' and D' denote cost of affine-coordinate addition and doubling, respectively, on the GLS curve. This section discusses assumptions and recent formulae improvements relative to the table, and highlights computational and practical differences between the curves and methods.

Table 3. Estimates for point multiplication on random curves $E/\mathbb{F}_{2^n}$ and GLS curves $E'/\mathbb{F}_{2^{2\ell}}$. Multiplication costs for $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^{2\ell}}$ are denoted by m and M, respectively. Inversion cost in $\mathbb{F}_{2^{2\ell}}$ is denoted by I. Costs for multiplication by curve parameter b (or $\sqrt{b}$) for each curve are denoted by b and B, estimated at $m/2$ and $M/3$; ψ denotes the cost of applying the endomorphism. The costs in field multiplications use $w = 5$, $I = 6M$, $\psi = M$, and scalar bitlengths 256 and 253 for E and E', resp.

		Costs for Point Multiplication	
Operation	Evaluation	Double-and-add	Halve-and-add
Random curve $E/\mathbb{F}_{2^n}$ with $D = 2m + 2b,\ A = 8m,\ H = 5/3m$			
kP	pre/post[a]	$D + (2^{w-2} - 1)A$	$D + (2^{w-1} - 2)A$
	per-bit	$D + \frac{1}{w+1}A$	$H + \frac{1}{w+1}(A + m)$
	$\mathbb{F}_{2^n}$-mult, $n = 257$	1168	926
GLS curve $E'/\mathbb{F}_{2^{2\ell}}$ with $D = 2M + 2B,\ D' = A' = 2M + I,\ H = 5/3M$			
kP	pre/post	$D' + (2^{w-2} - 1)A'$	
	per-bit	$D + \frac{1}{w+1}(A' + M)$	$H + \frac{1}{w+1}(A' + M)$
	$\mathbb{F}_{2^{2\ell}}$-mult, $\ell = 127$	1118	865
$k_1P + k_2\psi(P)$	pre/post	$D' + (2^{w-2} - 1)A' + 2^{w-2}\psi$	
	per-bit	$(D + \frac{2}{w+1}(A' + M))/2$	$(H + \frac{2}{w+1}(A' + M))/2$
	$\mathbb{F}_{2^{2\ell}}$-mult, $\ell = 127$	789	662

[a] Halve-and-add post-evaluation is multiple-accumulator fixup for right-to-left method.

5.1. **Random curves.** Fong et al. [7] compared point multiplication via doubling-based and halving-based methods on random curves with $a = 1$, with operation count and experimental data strongly favouring halving. Since then, formula improvements due to Kim and Kim [20] offer a significant acceleration to point doubling in the scenario where halving applies, namely $a = 1$ with a modest amount of per-curve precomputation permitted.[3]

Table 4 summarizes the costs of point addition and doubling in López-Dahab coordinates with various formulae and parameters. The significance of the Kim and Kim result is that total multiplications for doubling remain unchanged from earlier formulae, but the number of multiplications by curve parameter b is now 2 rather than 1. Since these formulae do not accelerate typical halving-based methods, we are interested in a revised doubling vs. halving comparison when precomputation significantly reduces the cost of multiplication by b. The comparison against halving is not quite

[3] Halving applies to curves with $\text{Tr}(a) = 0$, however there is a penalty of at least 1/2 field multiplication per halving step [23, 22]. The doubling-based approach has a smaller "chaining penalty" with the $a = 0$ formulae of Bernstein et al. [4].

Table 4. Point operation cost for mixed-coordinate additions and projective doublings, where the projective point $(X : Y : Z)$ corresponds to the affine point $(X/Z, Y/Z^2)$. M and S denote the costs of field multiplication and squaring, resp., and a, b, $\sqrt{b}$ denote the cost of multiplication by the corresponding curve parameters.

	Addition	Doubling	
López, Dahab [27]	$9M + 4S + 1a$	$3M + 5S + 1b + 1a$	
King [21]	$9M + 5S + 1a^{\mathrm{a}}$		
Lim, Hwang [26]; Al-Daoud, Mahmod, Rushdan [1]	$8M + 5S + 1a^{\mathrm{b}}$		
Lange [25]		$4M + 4S + 1a$	
Kim, Kim [20]		$2M + 5S + 2b$	$a = 1^{\mathrm{c}}$
Bernstein, Lange, Farashahi [4]		$2M + 4S + 1b + 1\sqrt{b}$	$a = 1^{\mathrm{c}}$
		$2M + 5S + 1b + 1\sqrt{b}$	$a = 0^{\mathrm{d}}$

[a]Provides XZ and Z^2 for subsequent double. [b]Provides Z^2 for subsequent double.
[c]Assumes Z^2 chained. [d]Assumes XZ and Z^2 chained.

as straightforward as it appears, since there is a one-multiplication "penalty" on essentially every point addition due to conversion from λ-coordinates. However, the $a = 1$ formula for doubling does not pay this penalty, and this is the comparison that is most meaningful in the present context.

The counts in terms of field multiplication in Table 3 assume that field representations can be selected where square roots are inexpensive, in which case the halving step is dominated by a multiplication and the cost of the quadratic solver. In the modest-storage scenario with fields of interest in this paper, we assume that a point halving can be done in 5/3 the time of a field multiplication (so $H \approx \frac{5}{3}m$), and that multiplication by a constant with precomputation requires half the time of a general multiplication (so $b \approx \sqrt{b} \approx m/2$). Under these assumptions, Table 3 gives a factor 1.26 advantage to the halving-based approach on E. This is smaller than the factor 1.41 estimate in [7] where the López-Dahab doubling formulas were used without off-line precomputation (but halving was estimated at a higher two-multiplication cost).

5.2. **GLS curves.** The GLS curves $E'/\mathbb{F}_{2^{2\ell}} : Y^2 + XY = X^3 + a'X^2 + b$ are "small parameter" in the sense that $b \in \mathbb{F}_{2^\ell}$ and a' can be chosen arbitrarily subject to the trace requirement $\mathrm{Tr}'(a') = 1$. Compared to the random curves, multiplication by the corresponding curve parameters b or $\sqrt{b}$ is a smaller fraction of the cost of a general multiplication. We can choose a' so that multiplication by a' is inexpensive; however, the GLS curves are trace-1 but not $a' = 1$. The Kim and Kim formula for point doubling does not apply, but the techniques for $a = 0$ in the formula of Bernstein-Lange-Farashahi can be adapted as follows. To find $2P = (x_3, y_3)$ for $P = (x, y) \notin \{-P, \infty\}$, write the affine formulae as $x_3 = x^2 + b/x^2$ and $y_3 = x^2 + (\lambda + 1)x_3$ where $\lambda = x + y/x$. Then

$$y_3 = x^2 + \left(\frac{x^2 + y + x}{x}\right)\left(\frac{x^4 + b}{x^2}\right) = \frac{x^6 + x(y^2 + y + x)(x^4 + b)}{x^4}$$

and, as in (part of) Kim and Kim, the substitution $x^6 = (y^2 + xy + a'x^2 + b)^2$ from the curve equation is used and the result written in López-Dahab projective form (U_3, V_3, W_3). Straightforward arithmetic gives the $a = 0$ Bernstein et al. formulae [4] but with the addition of $a'(a'T_3 + S_3)$ in the assignment to V_3. This formulation is of interest only if multiplication by a' is inexpensive, which is the case for GLS curves since parameter a' can be chosen so that this multiplication is essentially free.

Unlike Kim and Kim, the "two multiplications by constants" $a = 0$ Bernstein et al. formulas (and those adapted here for GLS) chain the quantity XZ in point doubling. The consequence can be similar to the point-halving method, where a point addition has a "penalty" of an extra multiplication. There is also multiplication by both b and $\sqrt{b}$ in point doubling, although (as in Kim and Kim) this can be done via two multiplications by b at the cost of an extra squaring.

The GLV technique also has a precomputation penalty in the sense that there are now two inputs P and $\psi(P)$ for which precomputation is required. If width-w NAF methods are employed, then the basic issue is that precomputation is used less efficiently when interleaving k_1P and $k_2\psi(P)$. In practice, this penalty is small unless constraints limit the amount of precomputation to less than four points. Further, the precomputation involving $\psi(P)$ can be obtained from that for P since ψ is inexpensive compared to point addition. For suitable a', ψ can be essentially free, in which case the table for $\psi(P)$ can be omitted and the values obtained on-demand from the table for P.

The $\mathrm{Tr}'(a') = 1$ condition on the GLS curves is favourable to point halving techniques (as discussed for random curves). The GLV and point halving techniques can be combined, with scalar recoding performed as follows. Assume $r \approx 2^{2\ell}$ and let $k' = 2^\ell k \bmod r$. Find $k' \equiv k_1' + k_2'\lambda \pmod{r}$ via the usual GLV technique, giving k_i' of expected length approximately ℓ. Then $k_i \leftarrow k_i'/2^\ell$ gives $k \equiv k_1 + k_2\lambda \pmod{r}$ with k_i in essentially the form we seek, other than a few terms which will involve doubling in point multiplication.

Finally, we note that the quadratic extension in GLS is attractive in the sense that operations in $\mathbb{F}_{2^{2\ell}}$ reduce to operations in $\mathbb{F}_{2^\ell}$. In particular, precomputation for the quadratic solver can require less storage than in the random curve case, and inversion in $\mathbb{F}_{2^{2\ell}}$ is reduced to three multiplications and an inversion in $\mathbb{F}_{2^\ell}$. This inversion is likely to be sufficiently inexpensive that point addition can be done in affine coordinates (and so halving-based methods can run entirely in affine).

5.3. **Additional accelerations.** Two types of accelerations are suggested by the GLS technique and the improved Kim and Kim doubling formulas that exploit precomputation. We briefly consider the effects of specializing curve parameters for random curves E and for GLS curves E' (where GLS has already forced a specialization of b, namely $b \in \mathbb{F}_{2^\ell}$). We also examine precomputation strategies that offer less dramatic acceleration than those involving curve parameters, but are similar in the sense that they can re-use tables across point operations.

Specialized curve parameters. Since the GLS curves E' are special in the sense that $b \in \mathbb{F}_{2^\ell}$, we consider specialized b for E and further specialization on E' to speed point multiplication. At the extreme are Koblitz curves where $b = 1$, in which case different techniques apply and will not be considered here. For non-Koblitz curves, selecting specialized b will reduce the cost of doubling while the cost of halving remains the same. If b is of low hamming weight, for example, then multiplications by b and $\sqrt{b}$ (given suitable field representation) in point doubling become inexpensive.

The overall improvement can be incremental if precomputation was already giving these multiplications at 1/2 or 1/3 the cost of a general multiplication in E and E', respectively, since the savings are diluted by point additions and other operations. However, if quite special b is acceptable and gives associated multiplication at, say, 1/8 the cost of general multiplication, then the estimates in Table 3 for E give 976 and 925 as the corresponding $\mathbb{F}_{2^n}$-multiplication counts for point multiplication on E via doubling and halving, resp., predicting that the two methods run at approximately the same time.

Re-use of per-point precomputation. Re-use of precomputation for field multiplication can be applied per-point, although the possible acceleration is much smaller than with modest amounts of per-curve precomputation involving b and $\sqrt{b}$. King [21] describes re-use of multiplication tables within a point double or point addition, and this technique is examined in more detail in [3] (where it is called "sequential multiplications") in the context of hyperelliptic curves where the arithmetic provides more opportunity to exploit the method.

King also describes a per-point precomputation strategy that applies in some point multiplication methods where there is a small set of points repeatedly accessed, as in left-to-right width-w NAF methods. The basic idea is to build tables for some of the field multiplications in point addition involving coordinates from the small set, since these coordinates will be used repeatedly. The cost of this precomputation is spread across the entire point multiplication rather than within a point operation, although admittedly the method is useful only for small w and the gains are incremental. Further, the right-to-left windowing method in [7] used with halve-and-add (when inversions are expensive) precludes this acceleration, since the small set of fixed points is replaced by multiple accumulators.

6. Experimental results

For a concrete comparison, we implemented point multiplication methods on two curves, each at roughly the 128-bit security level. The target environment is "workstation-class" systems which do not have severe memory constraints. In this section, we describe the curves, implementation issues, development environment, and give timings.

We chose curve and field parameters to be compatible with the 128-bit security requirement and with the methods of interest. For random curves we chose $n = 257$, and for GLS curves we could not resist selecting $\ell = 127$; if concerns about the security of elliptic curves over $\mathbb{F}_{2^{254}}$ remain (see §4), then $\ell = 131$ would be a suitable alternative. For the random curve E, point halving and the best doubling formulas desire $a = 1$, and we chose b pseudo-randomly so that $\#E(\mathbb{F}_{2^{257}}) = 2r$ with r prime. The reduction polynomials were selected so that both reduction and square roots would be at low cost. The following table summarizes the selections.[4]

Curve	a	b	Base Field	Extension	$\sqrt{x}$
$E/\mathbb{F}_{2^n}$, $n = 257$	1	$\in \mathbb{F}_{2^{257}}$	$\mathbb{F}_2[x]/(x^{257}+x^{41}+1)$	—	$x^{129}+x^{21}$
$E'/\mathbb{F}_{2^{2\ell}}$, $\ell = 127$	u	$\in \mathbb{F}_{2^{127}}$	$\mathbb{F}_2[x]/(x^{127}+x^{63}+1)$	$\mathbb{F}_{2^{127}}[u]/(u^2+u+1)$	$x^{64}+x^{32}$

For GLS, the choice $a' = u$ (compatible with the trace requirement) makes multiplication with a' very inexpensive (comparable to selecting $a \in \{0, 1\}$ for random curves). The corresponding s in GLS (see §3.1) satisfies $s^q + s = u + 1$ and so ψ is essentially free. Curve parameter $b \in \mathbb{F}_{2^{127}}$ was chosen so that $\#E'(\mathbb{F}_{2^{254}}) = 2r$ where r is prime.

6.1. Implementation.

The field and elliptic curve code was written in the C programming language, except for a small fragment in assembly to assist in finding the degree of a polynomial (for inversion). The big-number arithmetic required in the GLV method (to recode the scalar as $k \equiv k_1 + k_2\lambda \pmod{r}$) and in halving-based methods (to recode the scalar in base 1/2) is via the OpenSSL libraries.[5] There are divisions required in GLV recoding, but these can be done per-curve so that only multiplications and shifts are involved per-scalar. Recoding for halving involves a single per-scalar modular reduction. Regardless, these operations are a minor portion of the overall cost of scalar multiplication.

[4] There are polynomials $x^{127} + r(x)$ giving two-term $\sqrt{x}$ where r has smaller degree than in $f(x) = x^{127} + x^{63} + 1$; we chose f since there are powers differing by a multiple of 64, and the higher degree of r is not a significant factor in our environment. The root also is of advantageous form.

[5] The library at http://www.openssl.org is used in the communications tool OpenSSH, for example. Here, we are using only the big-number subset.

Field multiplication. Base field multiplication is via comb methods [28]. Briefly, a common case of combing calculates $c \cdot d$ with a single table of precomputation containing $p \cdot d$ for polynomials p of degree less than w for some small w (e.g., $w = 4$). The words of c are then "combed" w bits at a time to select the appropriate precomputed value to add at the desired location of the accumulator. These have been among the fastest methods for multiplication on general-purpose processors, and can be combined with shallow-depth Karatsuba-like techniques to control code expansion. However, going from 32- to 64-bit code can reduce the advantage of combing over the Karatsuba-down-to-word-multiplication used, for example, in the well-known MIRACL library [35].[6] Nonetheless, combing appears to be fastest in our environment, and we combed on field elements with comb width 4. For 32- and 64-bit code, elements in $\mathbb{F}_{2^{257}}$ can be represented in 8 or 4 words along with an extra bit. Multiplication suffers only a small penalty by handling this bit separately.

Elements in the extension field $\mathbb{F}_{2^{254}} = \mathbb{F}_{2^{127}}[u]/(u^2 + u + 1)$ are represented as a pair $(a_0, a_1) = a_0 + a_1 u$ of $\mathbb{F}_{2^{127}}$-elements, and extension field multiplication requires three multiplications in $\mathbb{F}_{2^{127}}$ via Karatsuba. Squaring requires two $\mathbb{F}_{2^{127}}$-squarings. We expect similar multiplication times for $\mathbb{F}_{2^{254}}$ and $\mathbb{F}_{2^{257}}$, although the practical comparison is not as clear as the mathematics indicates. Roughly speaking, comb multiplication looks best when the machine register size is small compared to the number of words representing a field element. Depending on platform and n, it is possible that combing is fastest on field elements rather than via shallow-depth Karatsuba-like techniques combined with combing (essentially the technique for the extension field).

We used combing with windows of width 8 for multiplications by constants (b and $\sqrt{b}$). This is double the window size used for general multiplication, and the precomputation of 256 elements (per constant) is off-line. Hence, the run time is expected to be approximately half the time of a general multiplication. A possible downside of combing is that compilers can be rather sensitive to the precise form of the code; e.g., some compilers do not optimize arrays as well as scalars, even when indices are known at compile-time. As a platform-dependent example, array dimensions and access order in a table-lookup method such as comb can mean 10% overall difference in multiplication due to the cost of addressing.

Inversion. Inversion is via a Euclidean-algorithm variant, and only limited optimization was performed. An assembly language fragment for Pentium-like systems was used to exploit a bitscan instruction (*bsr*) useful for speeding degree calculations. Inversion in $\mathbb{F}_{2^{254}}$ was performed by elementary matrix methods costing three $\mathbb{F}_{2^{127}}$-multiplications and an inversion in $\mathbb{F}_{2^{127}}$.

Solving quadratics. Finding solutions to $\lambda^2 + \lambda = c$ for trace-0 c in $\mathbb{F}_{2^{257}}$ was done via Algorithm 3.86 of [16] with 4-bit lookups in accumulation of half-traces. This technique is easier to implement for a given field than the small-table methods illustrated in [7], and uses less storage and has lower accumulation cost than the method used in [2]. The reduction polynomial $x^{257} + x^{41} + 1$ gives especially clean and efficient code, although this choice builds the table from 84 half-trace computations rather than the 71 elements specified by this algorithm for $x^{257} + x^{12} + 1$ (but the latter polynomial increases the cost of roots). The extra-digit storage-and-performance penalty noted for multiplication is avoided in the quadratic solver by not tracking the constant term.

Solutions in $\mathbb{F}_{2^{254}}$ are found by reduction to $\mathbb{F}_{2^{127}}$ as follows. For $c = c_0 + c_1 u$ with $c_i \in \mathbb{F}_{2^{127}}$, solve $\lambda^2 + \lambda = c_1$ to get λ_1, and then obtain λ_0 from $\lambda^2 + \lambda = c'_0 + \mathrm{Tr}(c'_0)$ where $c'_0 = c_0 + \lambda_1^2 = c_0 + c_1 + \lambda_1$. The solution is then $\lambda_0 + (\lambda_1 + \mathrm{Tr}(c'_0))u$.

[6] In personal communication, Julio López reports comb variants that can better exploit certain processor characteristics; in particular, his timings on an AMD Athlon64 are 35% faster on $\mathbb{F}_{2^{1223}}$ than timings from an Opteron in [15].

Table 5. Timings (in 10^3 cycles) on a 3.16 GHz Xeon with Sun C 5.9. Field operation "const" denotes multiplication by a constant with precomputation, and "solve" finds λ given trace-0 c so that $\lambda^2 + \lambda = c$. "D&A" is width-5 NAF double-and-add. "H&A" is width-5 NAF with right-to-left half-and-add on E [16, Algorithm 3.91] and affine additions on E'. The GLV decomposition is $kP = k_1P + k_2\psi(P)$ with width-5 NAF applied to k_1 and k_2.

Base field					Extension		kP		$k_1P + k_2\psi(P)$	
mul	const	inv	root	solve	mul	inv	D&A	H&A	D&A	H&A
Random curve $E/\mathbb{F}_{2^{257}}$, $\mathbb{F}_{2^{257}} = \mathbb{F}_2[x]/(x^{257}+x^{41}+1)$										
0.71	0.48	10.8	0.10	0.49	—	—	1057	790	—	—
GLS curve $E'/\mathbb{F}_{2^{254}}$, $\mathbb{F}_{2^{254}} = \mathbb{F}_{2^{127}}[u]/(u^2+u+1)$, $\mathbb{F}_{2^{127}} = \mathbb{F}_2[x]/(x^{127}+x^{63}+1)$										
0.27	0.15	3.4	0.06	0.21	0.87	4.6	1122	758	822	584

Square-root computations are inexpensive with the choice of reduction polynomials f. Coefficients corresponding to even and odd powers of x were extracted simultaneously by 8-bit lookup, and then $\sqrt{c} = \sum c_{2i}x^i + \sqrt{x}\sum c_{2i+1}x^i$. Since the degree of $\sqrt{x}$ is at most $(\deg f + 1)/2$ in our case, no reduction is required and finding a square root is less expensive than a squaring.

6.2. **Performance comparisons.** Timings in Table 5 were obtained on a Sun X4150, a 64-bit system built on the Intel Xeon processor (model 5460, 3.16 GHz). Roughly speaking, this processor family includes the Intel Core2 and the Athlon64 and Opteron from AMD, with similar instruction sets and timings. The compiler is Sun C 5.9, producing 64-bit objects.

The point multiplication times include the cost of scalar recoding ($k \equiv k_1 + k_2\lambda \pmod r$ in GLV, and base 1/2 form of k in halving). A sequence of pseudo-random elements and scalars was used in the timings to preclude effects such as branch prediction that would not apply outside of the laboratory. As expected, field multiplication times for $\mathbb{F}_{2^{257}}$ and $\mathbb{F}_{2^{254}}$ are roughly comparable. If these times were the same, then Table 3 would predict that E' is faster than E for point multiplication in the non-GLV timings. The experimental data shows that E does better than predicted in this comparison, and the main reason is that the comb multiplier is more efficient there than on the subfield $\mathbb{F}_{2^{127}}$.

The GLV technique provides an acceleration of 27% and 23% for doubling- and halving-based methods on E'; as expected, the combination of GLV decomposition and point halving gives the best time. A comparison of interest is between E' and E, and this is the comparison for prime fields in [10] where the acceleration from GLV is 14% on a 32-bit Pentium 4 and 43% on an 8-bit Atmel; a later draft included 64-bit code on an Intel Core2, with a 7% acceleration. The corresponding comparison for point halving methods in our case is 26%. While the comparison against E may indeed be the "right" one, it comes with a caveat that the measurement is polluted by differences in field multiplication times, a portion of which are due to programmer talent or lack thereof and vary by processor. A significant difference between the case for curves over prime fields from those over binary fields is the cost ratio of point addition to doubles. As rough approximations in the scenarios of interest, the ratio in the prime field case is 11/8, while the ratio for binary fields is 8/4. Since the GLV technique reduces doubles but not additions, the opportunity for improvement is apparently larger for prime fields.[7]

[7] The explanation in [10] for the lower-than-expected improvement on 32-bit code involves overheads when field elements are only a few words, along with implementation decisions that do not favour smaller fields.

7. Conclusions

The GLS curves over binary fields offer significant acceleration via the GLV decomposition technique. The advantage extends to comparisons against random curves, and with methods based on point halving. Recent improvements to doubling formulae have narrowed the performance gap between the doubling- and halving-based techniques, although halving retains a significant advantage provided a threshold amount of per-field precomputation is available for the quadratic solver. On the other hand, if fairly specialized curve parameters are acceptable, then the analysis shows that the new formulae can eliminate most of the advantage of halving.

References

[1] E. Al-Daoud, R. Mahmod, M. Rushdan and A. Kilicman, "A new addition formula for elliptic curves over $GF(2^n)$", *IEEE Transactions on Computers*, 31 (2002), 972–975.

[2] R. Avanzi, "Another look at square roots (and other less common operations) in fields of even characteristic", *Selected Areas in Cryptography – SAC 2007*, Lecture Notes in Computer Science, 4876 (2007), 138–154.

[3] R. Avanzi and N. Thériault, "Effects of optimizations for software implementations of small binary field arithmetic", *Arithmetic of Finite Fields – WAIFI 2007*, Lecture Notes in Computer Science, 4547 (2007), 69–84.

[4] D. Bernstein, T. Lange and R. Farashahi, "Binary Edwards curves", Cryptology ePrint Archive: Report 2008/171, 2008. Available from http://eprint.iacr.org/2008/171.

[5] C. Diem and E. Thomé, "Index calculus in class groups of non-hyperelliptic curves of genus three", *Journal of Cryptology*, to appear.

[6] A. Enge and P. Gaudry, "A general framework for subexponential discrete logarithm algorithms", *Acta Arithmetica*, 102 (2002), 83–103.

[7] K. Fong, D. Hankerson, J. López and A. Menezes, "Field inversion and point halving revisited", *IEEE Transactions on Computers*, 53 (2004), 1047–1059.

[8] S. Galbraith, "Constructing isogenies between elliptic curves over finite fields", *LMS Journal of Computation and Mathematics*, 2 (1999), 118–138.

[9] S. Galbraith, F. Hess and N. Smart, "Extending the GHS Weil descent attack", *Advances in Cryptology – EUROCRYPT 2002*, Lecture Notes in Computer Science, 2332 (2002), 29–44.

[10] S. Galbraith, X. Lin and M. Scott, "Endomorphisms for faster elliptic curve cryptography on general curves", Cryptology ePrint Archive: Report 2008/194, 2008. Available from http://eprint.iacr.org/2008/194.

[11] R. Gallant, R. Lambert and S. Vanstone, "Faster point multiplication on elliptic curves with efficient endomorphisms", *Advances in Cryptology – CRYPTO 2001*, Lecture Notes in Computer Science, 2139 (2001), 190–200.

[12] P. Gaudry, F. Hess and N. Smart, "Constructive and destructive facets of Weil descent on elliptic curves", *Journal of Cryptology*, 15 (2002), 19–46.

[13] S. Gueron and M. Kounavis. "Carry-less multiplication and its usage for computing the GCM mode", white paper, Intel Corporation, 2008. Available from http://softwarecommunity.intel.com/articles/eng/3787.htm.

[14] S. Gueron and M. Kounavis. "A technique for accelerating characteristic 2 elliptic curve cryptography", In *Fifth International Conference on Information Technology: New Generations (ITNG 2008)*, IEEE Computer Society, 2008, 265–272.

[15] D. Hankerson, A. Menezes, and M. Scott, "Software implementation of pairings", draft chapter, 2008.

[16] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2003.

[17] F. Hess, "Generalising the GHS attack on the elliptic curve discrete logarithm problem", *LMS Journal of Computation and Mathematics*, 7 (2004), 167–192.

[18] I. Iijima, K. Matsuo, J. Chao and S. Tsujii, "Construction of Frobenius maps of twists elliptic curves and its application to elliptic scalar multiplication", *Proceedings of the 2002 Symposium on Cryptography and Information Security – SCIS 2002*, Japan, 2002.

[19] D. Jao, S. Miller and R. Venkatesan, "Do all elliptic curves of the same order have the same difficulty of discrete log?", *Advances in Cryptology – ASIACRYPT 2005*, Lecture Notes in Computer Science, 3788 (2005), 21–40.

[20] K. Kim and S. Kim, "A new method for speeding up arithmetic on elliptic curves over binary fields", Cryptology ePrint Archive: Report 2007/181, 2007. Available from http://eprint.iacr.org/2007/181.

[21] B. King, "An improved implementation of elliptic curves over $GF(2^n)$ when using projective point arithmetic", *Selected Areas in Cryptography – SAC 2001*, Lecture Notes in Computer Science, 2259 (2001), 134–150.

[22] B. King and B. Rubin, "Improvements to the point halving algorithm", *Australasian Conference on Information Security and Privacy – ACISP 2004*, Lecture Notes in Computer Science, 3108 (2004), 262–276.

[23] E. Knudsen, "Elliptic scalar multiplication using point halving", *Advances in Cryptology – ASIACRYPT '99*, Lecture Notes in Computer Science, 1716 (1999), 135–149.

[24] N. Koblitz, "CM-curves with good cryptographic properties", *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science, 576 (1992), 279–287.

[25] T. Lange, "A note on López-Dahab coordinates", *Tatra Mountains Mathematical Publications*, 33 (2006), 75-81. Also available from http://eprint.iacr.org/2004/323.

[26] C. Lim and H. Hwang, "Speeding up elliptic scalar multiplication with precomputation", *Information Security and Cryptology '99*, Lecture Notes in Computer Science, 1787 (2000), 102–119.

[27] J. López and R. Dahab, "Improved algorithms for elliptic curve arithmetic in $GF(2^n)$", *Selected Areas in Cryptography – SAC '98*, Lecture Notes in Computer Science, 1556 (1999), 201–212.

[28] J. López and R. Dahab, "High-speed software multiplication in $\mathbb{F}_{2^m}$", *Progress in Cryptology – INDOCRYPT 2000*, Lecture Notes in Computer Science, 1977 (2000), 203–212.

[29] M. Maurer, A. Menezes and E. Teske, "Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree", *LMS Journal of Computation and Mathematics*, 5 (2002), 127–174.

[30] A. Menezes and M. Qu, "Analysis of the Weil descent attack of Gaudry, Hess and Smart", *Topics in Cryptology – CT-RSA 2001*, Lecture Notes in Computer Science, 2020 (2001), 308–318.

[31] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[32] B. Möller, "Algorithms for multi-exponentiation", *Selected Areas in Cryptography – SAC 2001*, Lecture Notes in Computer Science, 2259 (2001), 165–180.

[33] J. Muir and D. Stinson, "Minimality and other properties of the width-w nonadjacent form", *Mathematics of Computation*, 75 (2006), 369-384.

[34] R. Schroeppel, "Automatically solving equations in finite fields", US patent application No. 09/834,363, filed April 12, 2001.

[35] M. Scott, *MIRACL – Multiprecision Integer and Rational Arithmetic C Library*, http://www.computing.dcu.ie/~mike/miracl.html.

[36] J. Solinas, "Efficient arithmetic on Koblitz curves", *Designs, Codes and Cryptography*, 19 (2000), 195–249.

Department of Mathematics & Statistics, Auburn University, Auburn, Alabama 36849 USA
E-mail address: hankedr@auburn.edu

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada
E-mail address: kkarabin@uwaterloo.ca

Department of Combinatorics & Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada
E-mail address: ajmeneze@uwaterloo.ca