Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three

N.P. Smart, E.J. Westwood

Computer Science Department
University of Bristol
Woodland Road,
Bristol BS8 1UB,
United Kingdom.
{nigel, westwood}@cs.bris.ac.uk

Abstract. In this paper we investigate the efficiency of cryptosystems based on ordinary elliptic curves over fields of characteristic three. We look at different representations for curves and consider some of the algorithms necessary to perform efficient point multiplication. We give example timings for our operations and compare them with timings for curves in characteristic two of a similar level of security. We show that using the Hessian form in characteristic three produces a point multiplication algorithm under 50 percent slower than the equivalent system in characteristic two. Thus it is conceivable that curves in characteristic three, could offer greater performance than currently perceived by the community.

1 Introduction

Elliptic curve cryptography, proposed independently by Neal Koblitz [17] and Victor Miller [18] in 1985, continues to receive increasing commercial acceptance as can be seen by its inclusion in numerous standards such as ANSI X9.62 and X9.73, IEEE 1363, ISO 14888-3 and 15946 and NIST FIPS 186.2, For more information on these standards, see [1], [2], [3], [4], [5] and [6]. These standards provide carefully chosen curves and fields appropriate for secure implementations in both the large prime and binary cases. The standards reflect a huge research effort that has gone into finding such curves and underlying fields but despite this work some obvious areas of study still seem to have received relatively little attention.

In this investigation we look at ordinary elliptic curves defined over finite fields of characteristic three, a topic which has only seen a small amount of work when compared to that of curves over fields of even characteristic and large prime fields. Since we are only concerned with elliptic curves the term ordinary is equivalent to non-supersingular. One could ask why characteristic three deserves attention over characteristic five or seven? But curves over fields of characteristic three have certain properties that we can exploit to give fast computations without the need for excessive precomputations or storage requirements, keeping them suitable for constrained devices like smart cards or palm top organisers where such resources are at a premium. For a detailed study of algorithms in the prime and binary cases see [9] and [13], which cover details on both the curve arithmetic and that of the underlying finite fields.

Supersingular elliptic curves in characteristic three have recently received some attention as possible candidates for use in pairing-based based cryptosystems such as the Boneh-Franklin scheme [8]. These cryptosystems require efficient algorithms for computing some pairing map $e : E[n] \times E[n] \mapsto \mathbb{F}_{3^{6m}}$ as well as for point multiplication on the underlying supersingular curve. In this paper we are only concerned with the point multiplication operation on ordinary elliptic curves.

Some of the necessary tools for using characteristic three curves already exist. Until recently comparatively little work had been done on point counting in small odd characteristic. Luckily with the advent of Satoh's algorithm [20] we can compute group orders in characteristic three relatively cheaply.

In this paper we study the problem of point multiplication in two ways. The first is to take a curve in Weierstrass form and find a reasonably efficient point tripling formula that can be used in a triple and add style algorithm. This has the benefit that although the tripling operation may not be as fast as the one for doubling, the 3-adic expansion of a multiplier will be considerably shorter and thus potentially achieving some speed up. This can be a benefit since a single tripling operation can be accomplished more efficiently than a double and add combined.

The second way is to look for another form of the same curve which has a particularly fast doubling algorithm for characteristic three fields and then use the familiar double and add style algorithm. The Hessian form of an elliptic curve provides just such a doubling formula, since the doubling formula requires several cubings which are relatively fast in characteristic three.

The Hessian form of an elliptic curve has a long history, its group law was originally written down by Desboves in [11], see [12] for a mathematical overview of the Hessian form. It was introduced as a possible tool in elliptic curve cryptography in [21], where it was proposed to enable efficient parallel SIMD like field operations. It was then realised in [16] that the group law made it partially resistant to side channel analysis. However, both of these papers made the assumption that the curve was defined over a field of order q, with $q \equiv 2 \pmod 3$. They hence explicitly ruled out the characteristic three case.

The remainder of this paper is organised as follows. In Section 2 we recap on some basic definitions and notation, such as facts about characteristic three fields and coordinate systems on curves. In Section 3 we look at the two different models we will be considering for our elliptic curves, namely Weierstrass and Hessian. We discuss the properties we wish the curve to possess and indicate how the two forms are related. Section 4 explains in more detail the various methods of point multiplication for the Weierstrass and Hessian forms. Section 5 gives our experimental results and finally in Section 6 we give some conclusions.

It is hoped that this paper will fill a gap between the theory of implementations in prime fields and those in binary fields and show that characteristic three curves are a viable alternative to existing technology for discrete logarithm based elliptic curve cryptography.

2 Elliptic Curves over Fields of Characteristic Three

We will give some basic facts about elliptic curves over characteristic three fields since most of the cryptographic literature does not deal adequately with this case. It is essential for our purposes that we can work efficiently in the field $\mathbb{F}_{3^m}$ if we are to use it in cryptographic applications. The papers [14] and [19] explain details of implementing characteristic three field arithmetic in both software and hardware. The main point to note for our purposes is that cubing a field element is very easy, using an analogue of the usual 'thinning algorithm' in characteristic two: For an element represented by the polynomial $g(z)$ in the field $\mathbb{F}_{3^m} = \mathbb{F}_3[z]/M(z)$ the cubing rule can be expressed as

$$g(z)^3 = g(z^3) \pmod{M(z)}.$$

Multiplying and squaring field elements are of similar complexity to each other and are not too expensive but more so than cubing. Typically one can make the cubing operation at least ten times faster than a single multiplication or squaring. Addition and subtraction are as usual virtually free. We also remind the reader that in $\mathbb{F}_{3^m}$ the map $\alpha \mapsto \alpha^3$ is a permutation and therefore every element has a unique cube root.

The elliptic curves over $\mathbb{F}_{3^m}$ can be broken down into two categories, ordinary elliptic curves and supersingular elliptic curves. We shall only be interested in ordinary curves, which have the Weierstrass form

$$y^2 = x^3 + ax^2 + c \text{ with both } a \text{ and } c \text{ non zero.}$$

or equivalently (for curves with a point of order three) the Hessian form

$$x^3 + y^3 + 1 = Dxy \text{ with } D \neq 0.$$

The conditions on the coefficients ensure that the curves are nonsingular.

3 The Weierstrass and Hessian Forms

The Weierstrass Form: Affine coordinates (x, y) are the simplest when considering point addition, doubling and tripling and we have the following simple formulae for the characteristic three case.
Affine Addition: $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ where we set

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

and then

$$x_3 = \lambda^2 - a - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

Affine Doubling: $2(x, y) = (x_2, y_2)$ where we set

$$\lambda = \frac{ax}{y}$$

and then

$$x_2 = \lambda^2 - a + x$$
$$y_2 = \lambda(x - x_2) - y$$

Affine Tripling: $3(x, y) = (x_3, y_3)$ where

$$x_3 = \frac{(x^3 + c)^3 - a^3 c x^3}{a^2 (x^3 + c)^2}$$
$$y_3 = \frac{y^9 - a^3 y^3 (x^3 + c)^2}{a^3 (x^3 + c)^3}$$

Before proceeding we note that these naive formulae all require field inversions, which is a relatively expensive operation compared to multiplication, squaring and addition. We can get around this problem by using projective coordinates (X, Y, Z). When working in projective coordinates we can calculate point additions, doublings and tripling without costly field inversions. A projective point (X, Y, Z) can be made into an affine point (x, y) at the end of a calculation, if required, at the cost of one field inversion and some multiplies.

There are several different types of projective coordinates and the relationship between (x, y) and (X, Y, Z) are as follows.

- Ordinary projective: $(x, y) = (X/Z, Y/Z)$
- Jacobian projective: $(x, y) = (X/Z^2, Y/Z^3)$
- López Dahab projective [15]: $(x, y) = (X/Z, Y/Z^2)$

It turns out that for curves in Weierstrass form the Jacobian representation appears to be more efficient. To obtain the curve in Jacobian projective coordinates we make the substitution $x = X/Z^2$ and $y = Y/Z^3$. The equation of the curve can then be expressed as

$$Y^2 = X^3 + aX^2Z^2 + cZ^6.$$

To simplify some of the curve operations it will be convenient to set $a = 1$. Insisting on having $a = 1$ eliminates half of the curves in characteristic three, and is akin to the assumption for curves over large prime to have the form

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

Lemma 1. *A curve has a point of order three if and only if it can be written in the form*

$$y^2 = x^3 + x^2 + c.$$

Proof. Suppose we have a curve with a point of order three given by

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

Then using the above formulae for the affine tripling law we see that we must have $a(x^3 + c) = 0$ for a point of order three (x, y). Since we must have $a \neq 0$ for the curve to be non-singular we see that $x^3 = -c$.

Since we are in the field $\mathbb{F}_{3^m}$ we know any $-c$ has a (unique) cube root in the field, call this l so that $l^3 = -c$. This means that the point (l, y) is of order three provided that y satisfies the equation of the curve, that is

$$y^2 = l^3 + al^2 + c = al^2.$$

Notice, the equation $y^2 = al^2$ has a solution if and only if a is a quadratic residue in the field. Hence, a point of order three exists if and only if a is a quadratic residue in the field.

Suppose a is a quadratic residue, then there exists a u such that $u^2 = a$. Now we can make the substitution $x = u^2\overline{x}$, $y = u^3\overline{y}$ giving

$$\begin{aligned} u^6\overline{y}^2 &= u^6\overline{x}^3 + au^4\overline{x}^2 + c \\ &= u^6\overline{x}^3 + u^6\overline{x}^2 + c \end{aligned}$$

and dividing by u^6 gives

$$\overline{y}^2 = \overline{x}^3 + \overline{x}^2 + c'.$$

The converse is trivial.

Q.E.D.

From now on we will only consider Weierstrass equations of the form

$$y^2 = x^3 + x^2 + c.$$

Having the elliptic curve in this slightly simplified form speeds up the addition, doubling and tripling operations since multiplication by a is now trivial but this simplification happens at the expense of one bit of security. If we use a curve of the form $y^2 = x^3 + x^2 + c$ then we know it has a point of order three and hence the order of the group is divisible by three. When using a curve of this form we should check that it has group order $3 \times p$ where p is a prime and then work in the subgroup of size p.

The Hessian Form: The Hessian form of an elliptic curve is less common in the literature, but it has been considered by a number of authors in various situations [10], [11], [16] and [21]. If an elliptic curve has a point of order three, in which case we can assume the curve can be written in a Weierstrass form with $a = 1$, then the curve can also be written as

$$x^3 + y^3 + 1 = Dxy$$

in affine coordinates and if we make the substitution $x = X/Z$ and $y = Y/Z$ the equation can be expressed as $X^3 + Y^3 + Z^3 = DXYZ$ in ordinary projective coordinates.

The points on the curve

$$X^3 + Y^3 + Z^3 = DXYZ$$

again form an additive group. The zero of the group law is given by $(1, -1, 0)$. The two points of order three are given by $(0, 1, -1)$ and $(1, 0, -1)$. If $P = (x_1, y_1, z_1)$ and $Q = (x_2, y_2, z_2)$, we define

$$\begin{aligned} -P &= (y_1, x_1, z_1), \\ P + Q &= (x_3, y_3, z_3), \end{aligned}$$

where

$$\begin{aligned} x_3 &= y_1^2 x_2 z_2 - y_2^2 x_1 z_1, \\ y_3 &= x_1^2 y_2 z_2 - x_2^2 y_1 z_1, \\ z_3 &= z_1^2 y_2 x_2 - z_2^2 y_1 x_1. \end{aligned}$$

The formulae for point doubling are given by $[2]P = (x_2, y_2, z_2)$ where

$$\begin{aligned} x_2 &= y_1(z_1^3 - x_1^3), \\ y_2 &= x_1(y_1^3 - z_1^3), \\ z_3 &= z_1(x_1^3 - y_1^3). \end{aligned}$$

These formulae apply in all characteristics as well as characteristic three.

We can easily pass from a Weierstrass form

$$y^2 = x^3 + x^2 + c$$

to the Hessian form, and vice-versa, as follows. We set as before l to be the unique field element satisfying $l^3 = c$. The element l can be found by setting $l = c^{3^{m-1}}$, when the finite field is of order $q = 3^m$. Now taking a Weierstrass equation $y^2 = x^3 + x^2 + c$ and applying the coordinate transformation

$$\begin{aligned} x &= l(\overline{x} + \overline{y}), \\ y &= l(\overline{x} - \overline{y}) \end{aligned}$$

gives the required form

$$\overline{x}^3 + \overline{y}^3 + 1 = D\,\overline{x}\,\overline{y}$$

where $D = -l^{-1}$.

4 Comparing the representations

Before examining point multiplication we first need to discuss the best way to perform point addition, doubling and tripling. Tables 1 and 2 describe the exact operation counts for different coordinate systems on our two models for an elliptic curve. We ignore the cost of additions and count the number of cubes (C), squares (S), multiplies (M) and inversions (I).

It should be noted that the cost of a cubing operation is significantly lower than that of a square or multiply, and that theoretically a square should be slightly faster than a multiply.

The Jacobian and López Dahab coordinates are sometimes called weighted projective coordinates. After an evaluation to see which was the most efficient in characteristic three fields it was decided that Jacobian projective coordinates would be used for curves in Weierstrass form and ordinary projective coordinates for curves in Hessian form.

As usual we use the term 'mixed addition' to refer to a point addition where one point is affine and the other point is in a projective representation. The result of the mixed addition operation will in general not be an affine point. Such mixed addition algorithms are often used in point multiplication algorithms, see [9] and [13].

In Table 1 we present the operation counts for the various forms of coordinate systems on a curve in Weierstrass form with $a = 1$. In Table 2 we do the same for the Hessian form. Note that tripling in the Hessian form is not efficient so we do not cover this operation. In addition we only cover the projective coordinate system for the Hessian form since this leads to the most efficient implementation. The fastest way we could find to compute a point triple was to perform a double and then an add, hence we do not give details of point tripling costs in Table 2.

Table 1. Operation count for a curve in Weierstrass form with $a = 1$.

Operation	Coordinate System	Cost
Addition	Affine	0C + 1S + 2M + 1I
Doubling	Affine	0C + 1S + 2M + 1I
Mixed Addition	Projective	1C + 2S + 9M
Mixed Addition	Jacobian	2C + 3S + 7M
Mixed Addition	López Dahab	0C + 3S + 10M
Doubling	Projective	3C + 0S + 6M
Doubling	Jacobian	3C + 2S + 6M
Doubling	López Dahab	2C + 4S + 7M
Tripling	Affine	5C + 2S + 4M + 1I
Tripling	Projective	5C + 2S + 7M
Tripling	Jacobian	4C + 1S + 6M
Tripling	López Dahab	5C + 3S + 10M

Table 2. Operation count for a curve in Hessian form.

Operation	Coordinate System	Cost
Addition	Affine	0C + 0S + 11M + 1I
Doubling	Affine	2C + 0S + 5M + 1I
Mixed Addition	Projective	0C + 0S + 10M
Doubling	Projective	3C + 0S + 3M

4.1 Point Multiplication

We wish to know if characteristic three arithmetic provides an efficient way of calculating point multiplications on ordinary elliptic curves, since this is the fundemental operation on which cryptosystems are built. The usual way of doing this for curves over other fields is based on either a double-and-add method or some efficient endomorphism. We will not deal with the endomorphism method here since we are interested in general curves. We have two methods to test out:

- One is to use a double-and-add type method with the Hessian form since this has a very fast doubling operation in characteristic three fields.
- The other is to use a triple-and-add type method with Weierstrass curves as this exploits the fact that, given an exponent k, the 3-adic expansion of k is shorter than the binary (2-adic) expansion. Suppose k is an n bit number, then $k < 2^n < 3^m$ for some m. Picking the smallest such m we have that k is only m digits long in a base three expansion and $m = \lfloor n \log_3 2 \rfloor + 1$. Since $\log_3 2 \approx 0.63093$, the 3-adic expansion is significantly shorter.

We will use various methods of point multiplication with varying levels of optimization, but they are all based on the two standard algorithms. The following describe algorithms with window width one, but sliding signed window methods follow as usual from these basic algorithms. For an overview of these see [7] which gives details on the window methods based on binary algorithm. The corresponding algorithms for the ternary methods are obvious extensions of these.

Double and Add Algorithm
INPUT: Point P, an exponent $k = (a_n, \dots, a_2, a_1)_2$,
i.e. k is written in base 2 and $a_i \in \{0, 1\}$.
OUTPUT: Elliptic curve point $Q = [k]P$.

1. $Q = \mathcal{O}$.
2. for i = n to 1 do:
3. $\quad Q \leftarrow [2]Q$,
4. $\quad$ if $a_i = 1$ then $Q \leftarrow Q + P$.
5. Return Q.

Triple and Add Algorithm
INPUT: Point P, an exponent $k = (a_m, \dots, a_2, a_1)_3$,
i.e. k is written in base 3 and $a_i \in \{0, 1, 2\}$.
OUTPUT: Elliptic curve point $Q = [k]P$.

1. Precompute $[2]P$.
2. $Q = \mathcal{O}$.
3. for i = m to 1 do:
4. $\quad Q \leftarrow [3]Q$,
5. $\quad$ if $a_i = 1$ then $Q \leftarrow Q + P$,
6. $\quad$ else if $a_i = 2$ then $Q \leftarrow Q + [2]P$.
7. Return Q.

We will at this point examine how large the ratio needs to be between the time to compute an inversion and the time to compute a multiplication before the affine version becomes more efficient. We assume the time for a squaring is equal to the time for a general multiplication, and we assume the time for a cubing is negligible. This later assumption is not quite true but making it will make our analysis easier and will not effect the overall conclusion that much.

We first consider Weierstrass form: A nonary multiplication routine requires m triplings and $m/2+8$ additions. Hence, we have the following operation counts:

Affine Only	$(6m + 3m/2 + 24)M + (m + m/2 + 8)I$
Mixed	$(7m + 10m/2 + 24)M + 8I$

If we write $1I = rM$ then affine will be faster than mixed when

$$(15m + 48) + (3m + 16)r < (24m + 48) + 16r.$$

In other words

$$3mr < 9m,$$

hence we must have $r < 3$ to make affine coordinates more efficient.

We now consider the case of the Hessian form: A window multiplication routine of width five requires n doublings and $n/6 + 7$ additions. Hence, we have the following operation counts:

Affine Only	$(5n + 11n/6 + 77)M + (n + n/6 + 7)I$
Mixed	$(3n + 10n/6 + 77)M + 7I$

Again we write $1I = rM$ then affine will be faster than mixed when

$$41n + 7nr < 28n.$$

In other words

$$7nr < -13n,$$

hence, no matter how fast we make inversion, the mixed scheme will always beat the affine system.

We finally should consider the difference between the Weierstrass mixed nonary algorithm and the Hessian mixed binary window method. Here, we do need to take into account the cost of a cubing operation. The Weierstrass form will be faster than the Hessian form if

$$(4m+m)C + (7m+5m+24)M + 8I < 3nC + (3n+10n/6+77)M + 7I.$$

If we write $1I = rM$ and $1M = sC$ then Weierstrass will be faster than Hessian when

$$5m + (17m/2+24)s + 8rs < 3n + (28n/6+77)s + 7rs.$$

But we have $m \approx 0.64n$ and so our inequality becomes

$$0.2n + 0.77ns + rs < 53s.$$

Hence, asymptotically the Hessian method will always outperform the Weierstrass method, no matter what the relationship between the performance of multiplication, inversion and cubing.

5 Timings

In this section we give timings, of the various algorithms, for our C++ implementation on a Sparc Ultra 10. We also give timings for point exponentiations on an even characteristic curve with a similar security parameter so as to act as a benchmark.

For the curve in characteristic three we used the curve defined over

$$\mathbb{F}_{3^{97}} = \mathbb{F}_3[\theta]/(\theta^{97}+\theta^{12}+2)$$

given by

$$c = \texttt{0x5C6A21D1BF0967068295B8EAA7253DD2BD7A72}.$$

The above hexadecimal form of the field element is obtained in the standard way. The curve

$$E : y^2 = x^3 + x^2 + c$$

has group order

$$\begin{aligned}\#E &= 19088056323407827075424645001545815478748968937\\ &= 3 \times 6362685441135942358474881667181938492916322979,\end{aligned}$$

the last factor being a prime. We are grateful to D. Kohel for supplying an implementation of Satoh's algorithm [20] in characteristic three which enabled us to find this curve.

Table 3. Timings of Group Operations in Characteristic Three: $\mathbb{F}_{3^{97}}$

Form	Operation	Coordinates	Timing (in μs)
Weierstrass	Addition	Affine	346
Weierstrass	Double	Affine	291
Weierstrass	Triple	Affine	354
Weierstrass	Addition	Mixed	223
Weierstrass	Double	Mixed	153
Weierstrass	Triple	Mixed	134
Hessian	Addition	Affine	414
Hessian	Double	Affine	330
Hessian	Addition	Mixed	181
Hessian	Double	Mixed	62

In Table 3 we give the timings of our implementation in characteristic three. We give timings for the standard Weierstrass form and the Hessian form, for a curve of with $a = 1$ (i.e. 3 divides the group order).

In addition we give timings for both binary multiplication (in both affine and mixed coordinates) and a window based method for mixed coordinates, see Table 4. For the Weierstrass form we also present the timings for the nonary method.

For comparison with more traditional ECC implementations we present our timings in Table 5 for a similar sized elliptic curve over a field of characteristic two on the same computer. The results for characteristic two fields are included to give some idea of a comparison since the security of curves over $\mathbb{F}_{2^{163}}$ and $\mathbb{F}_{3^{97}}$ is approximately the same. Note that whilst $\log_2 3^{97} \approx 154$ the nearest standardized elliptic curve in characteristic two occurs for the field $\mathbb{F}_{2^{163}}$, [1].

Table 4. Timings in Characteristic Three: $\mathbb{F}_{3^{97}}$

Form	Method	Window length	Coordinates	Timing (in ms)
Weierstrass	Double and add	5	Affine	54.9
Weierstrass	Double and add	1	Mixed	62.8
Weierstrass	Double and add	5	Mixed	31.3
Weierstrass	Triple and add	2	Mixed	24.4
Hessian	Double and add	5	Affine	63.0
Hessian	Double and add	1	Mixed	65.6
Hessian	Double and add	5	Mixed	17.2

We conclude from the tables, that characteristic three curves do provide an alternative to curves over fields of even or large prime characteristic. Especially when we consider the Hessian form of an elliptic curve.

Table 5. Timings in Characteristic Two: $\mathbb{F}_{2^{163}}$

Form	Method	Window length	Coordinates	Timing (in ms)
Weierstrass	Double and add	5	Affine	27.6
Weierstrass	Double and add	1	Mixed	20.1
Weierstrass	Double and add	5	Mixed	12.7

6 Conclusion

We have demonstrated that Hessian arithmetic in characteristic three offers an alternative to traditional even characteristic and large prime variants of elliptic curve cryptography. We have also investigated the use of 3-adic expansions, which utilise the relatively cheap tripling operation in Weierstrass form. However, the benefit of 3-adic expansions is not as much as one would have hoped, and certainly not as efficient as Hessian arithmetic.

Since less time and research has been put into speeding up characteristic three field arithmetic we can hope to see these timings drop in the future providing further impetus to consider these curves as practical cryptographic tools when designing new encryption systems.

References

1. ANSI X9.62 Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999.
2. ANSI X9.63 Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Key Transport Protocols, working draft, October 2000.
3. IEEE 1363-2000 Standard Specifications for Public Key Cryptography, 2000.
4. ISO/IEC 14888-3 Information Technology - Security Techniques - Digital Signatures with Appendix - part 3: Certificate Based Mechanisms, 1998.
5. ISO/IEC 15946 Information Technology - Security Techniques - Cryptographic Techniques Based on Elliptic Curves, Committee Draft (CD), 1999.
6. National Institute of Standards and Technology Digital Signature Standard, FIPS Publication 186-2, February 2000.
7. I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, 1999.
8. D. Boneh and M. Franklin. Identity-based encryption form the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, Springer LNCS 2139, 213–229, 2001.
9. M. Brown, D. Hankerson, J. López and A.J. Menezes. Software implementation of NIST elliptic curves over prime fields. In *Topics in Cryptology – CT-RSA 2001*, Springer LNCS 2020, 250–265, 2001.
10. D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorisation tests. *Adv. in Appl. Math.*, **7**, 385–434, 1987.
11. A. Desboves. Résolution en nombres entiers et sous sa forme la plus générale, de l'équation cubique, homogéne á trois inconnue. *Nouv. Ann. de la Math.*, **5**, 545–579, 1886.

12. H.R. Frium. The group law on elliptic curves on Hesse form. Preprint, 2001.
13. D. Hankerson, J.L. Hernandez and A.J. Menezes. Software implementation of elliptic curve cryptography over binary fields. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, Springer LNCS 1965, 1–24, 2000.
14. K. Harrison, D. Page and N.P. Smart. Software implementation of finite fields of characteristic three. Preprint, 2002.
15. J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic in $GF(2^n)$. In *Selected Areas in Cryptography - SAC '98*, Springer-Verlag, LNCS 1556, 201–212, 1999.
16. M. Joye and J.-J. Quisquater. Hessian elliptic curves and side-channel attacks. *Cryptographic Hardware and Embedded Systems – CHES 2001*, Springer LNCS 2162, 402–410, 2001.
17. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, **48**, 203–209, 1987.
18. V. Miller. Uses of elliptic curves in cryptography. *Advances in Cryptology – CRYPTO '85*, Springer LNCS 218, 417–426, 1986.
19. D. Page and N.P. Smart. Hardware implementation of finite fields of characteristic three. To appear *Cryptographic Hardware and Embedded Systems – CHES 2002*,
20. T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.,*, **15**, 247–270, 2000.
21. N.P. Smart. The Hessian form of an elliptic curve. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, Springer LNCS 2162, 118–125, 2001.

7 Appendix: Group Laws

In this appendix we give the explicit formulae needed to perform the arithmetic operations on the curves. We present the formulae in a means more amenable to efficient implementation.

Table 6. Weierstrass Jacobian Addition

$O_1 = Z_1^2$	S
$O_2 = Z_2^2$	S
$O_3 = Z_1^3$	C
$O_4 = Z_2^3$	C
$O_5 = X_2 O_1$	M
$O_6 = X_1 O_2$	M
$O_7 = O_5 - O_6$	A
$O_8 = O_5 + O_6$	A
$O_9 = Y_2 O_3$	M
$O_{10} = Y_1 O_4$	M
$O_{11} = O_9 - O_{10}$	A
$O_{12} = Z_1 Z_2$	M
$O_{13} = O_7 O_{12} = Z_3$	M
$O_{14} = O_{12}^2$	S
$O_{15} = O_5 + O_6 + O_{14}$	2A
$O_{16} = O_7^2$	S
$O_{17} = O_{15} O_{16}$	M
$O_{18} = O_{11}^2$	S
$O_{19} = O_{18} - O_{17} = X_3$	A
$O_{20} = O_6 O_{16}$	M
$O_{21} = O_{20} - O_{19}$	A
$O_{22} = O_{11} O_{21}$	M
$O_{23} = O_7^3$	C
$O_{24} = O_{10} O_{23}$	M
$O_{25} = O_{22} - O_{24} = Y_3$	A
TOTAL	3C + 10M + 5S

Table 7. Weierstrass Mixed Addition ($Z_1 = 1$)

$O_1 = Z_2^2$	S
$O_2 = Z_2^3$	C
$O_3 = X_1 O_1$	M
$O_4 = X_2 - O_3$	A
$O_5 = X_2 + O_3$	A
$O_6 = Y_1 O_2$	M
$O_7 = Y_2 - O_6$	A
$O_8 = O_4 Z_2 = Z_3$	M
$O_9 = X_2 + O_3 + O_1$	2A
$O_{10} = O_4^2$	S
$O_{11} = O_9 O_{10}$	M
$O_{12} = O_7^2$	S
$O_{13} = O_{12} - O_{11} = X_3$	A
$O_{14} = O_3 O_{10}$	M
$O_{15} = O_{14} - O_{13}$	A
$O_{16} = O_7 O_{15}$	M
$O_{17} = O_4^3$	C
$O_{18} = O_6 O_{17}$	M
$O_{19} = O_{16} - O_{18} = Y_3$	A
TOTAL	2C + 7M + 3S

Table 8. Weierstrass Jacobian Double

$O_1 = Z^2$	S
$O_2 = O_1^3$	C
$O_3 = X^3$	C
$O_4 = cO_2$	M
$O_5 = O_3 + O_4$	A
$O_6 = Y^3$	C
$O_7 = YO_6$	M
$O_8 = O_1 O_4$	M
$O_9 = XO_5$	M
$O_{10} = -O_8 + O_9 = X_2$	A
$O_{11} = O_1^2$	S
$O_{12} = O_9 O_{11}$	M
$O_{13} = O_{12} - O_7 = Y_2$	A
$O_{14} = YZ = Z_2$	M
TOTAL	3C + 6M + 2S

Table 9. Weierstrass Jacobian Triple

$O_1 = X^3$	C
$O_2 = Y^3$	C
$O_3 = Z^3$	C
$O_4 = O_3^2$	S
$O_5 = cO_4$	M
$O_6 = O_4O_5$	M
$O_7 = O_1 + O_5$	A
$O_8 = O_1 - O_5$	A
$O_9 = O_5O_1$	M
$O_{10} = O_7^3$	C
$O_{11} = O_{10} - O_9 = X_3$	A
$O_{12} = O_7O_3 = Z_3$	M
$O_{13} = O_6O_8$	M
$O_{14} = O_{10} + O_{13}$	A
$O_{15} = O_2O_{14} = Y_3$	M
TOTAL	4C + 6M + S

Table 10. Hessian Projective Add

$O_1 = Y_1X_2$	M
$O_2 = X_1Y_2$	M
$O_3 = X_1Z_2$	M
$O_4 = Z_1X_2$	M
$O_5 = Z_1Y_2$	M
$O_6 = Z_2Y_1$	M
$O_7 = O_1O_6$	M
$O_8 = O_2O_3$	M
$O_9 = O_5O_4$	M
$O_{10} = O_2O_5$	M
$O_{11} = O_1O_4$	M
$O_{12} = O_6O_3$	M
$O_{13} = O_7 - O_{10} = X_3$	A
$O_{14} = O_8 - O_{11} = Y_3$	A
$O_{15} = O_9 - O_{12} = Z_3$	A
TOTAL	0C + 12M + 0S

Table 11. Hessian Mixed Addition

$O_1 = Y_1X_2$	M
$O_2 = X_1Y_2$	M
$O_3 = X_1Z_2$	M
$O_4 = Z_2Y_1$	M
$O_5 = O_1O_4$	M
$O_6 = O_2O_3$	M
$O_7 = X_2Y_2$	M
$O_8 = O_2Y_2$	M
$O_9 = O_1X_2$	M
$O_{10} = O_4O_3$	M
$O_{11} = O_5 - O_8 = X_3$	A
$O_{12} = O_6 - O_9 = Y_3$	A
$O_{13} = O_7 - O_{10} = Z_3$	A
TOTAL	0C + 10M + 0S

Table 12. Hessian Double

$O_1 = X^3$	C
$O_2 = Y^3$	C
$O_3 = Z^3$	C
$O_4 = O_3 - O_1$	A
$O_5 = O_2 - O_3$	A
$O_6 = O_1 - O_2$	A
$O_7 = YO_4 = X_2$	M
$O_8 = XO_5 = Y_2$	M
$O_9 = ZO_6 = Z_2$	M
TOTAL	3C + 3M + 0S