Efficient Pairing Computation on Genus 2 Curves in Projective Coordinates

Xinxin Fan[1], Guang Gong[1] and David Jao[2]

[1] Department of Electrical and Computer Engineering
[2] Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, N2L 3G1, CANADA
x5fan@engmail.uwaterloo.ca,
G.Gong@ece.uwaterloo.ca, djao@math.uwaterloo.ca

Abstract. In recent years there has been much interest in the development and the fast computation of bilinear pairings due to their practical and myriad applications in cryptography. Well known efficient examples are the Weil and Tate pairings and their variants such as the Eta and Ate pairings on the Jacobians of (hyper-)elliptic curves. In this paper, we consider the use of projective coordinates for pairing computations on genus 2 hyperelliptic curves over prime fields. We generalize Chatterjee *et. al.*'s idea of encapsulating the computation of the line function with the group operations to genus 2 hyperelliptic curves, and derive new explicit formulae for the group operations in projective and new coordinates in the context of pairing computations. When applying the encapsulated explicit formulae to pairing computations on supersingular genus 2 curves over prime fields, theoretical analysis shows that our algorithm is faster than previously best known algorithms whenever a field inversion is more expensive than about seventeen field multiplications. We also investigate pairing computations on non-supersingular genus 2 curves over prime fields based on the new formulae, and detail the various techniques required for efficient implementation.

Keywords: Genus 2 hyperelliptic curves, Tate pairing, Miller's algorithm, Projective coordinates, Efficient Implementation.

1 Introduction

Bilinear pairings were first introduced to cryptography by Menezes *et. al.* [26] and Frey and Rück [13] as a tool to attack instances of the discrete logarithm problem (DLP) on elliptic curves and hyperelliptic curves. Subsequently, Sakai *et. al.*'s non-interactive key distribution scheme [32] and Joux's tripartite Diffie-Hellman key agreement protocol [20] provided examples of positive usages of pairings. This use of pairings has inspired much research devoted to the design of cryptographic protocols with novel properties and the improvement of existing ones, with some classical examples being Identity Based Encryption [5] and short signatures [6]. Since pairing computations are generally the most important and expensive operation in any pairing-based cryptosystem, improving the speed of pairing computations has become an important issue in pairing-based cryptography.

Miller proposed the first algorithm [28] for computing the Weil pairing on elliptic curves. However, the Tate pairing shows better performance than that of the Weil pairing and therefore is widely used in practice. While many important techniques have been proposed to accelerate the computation of the Tate pairing and its variants on elliptic curves [2–4, 18], the subject of pairing computations on hyperelliptic curves is also receiving an increasing amount of attention. Choie and Lee [8] investigated the implementation of the Tate pairing on supersingular genus 2 hyperelliptic curves over prime fields. Later on, Ó hÉigeartaigh and Scott [17] improved the implementation of [8] significantly by using a new variant of Miller's algorithm combined with various optimization techniques. Duursma and Lee [10] presented a closed formula for the Tate pairing computation on a very special family of supersingular hyperelliptic curves. Barreto *et. al.* [2] generalized the results of Duursma and Lee and proposed the Eta pairing approach for efficiently computing the Tate pairing on supersingular genus 2 curves over binary fields. In particular, their algorithm leads to the fastest pairing implementation in the literature. In [25], Lee *et. al.* considered the Eta pairing computation on general divisors on supersingular genus 3 hyperelliptic curves with the form of $y^2 = x^7 - x \pm 1$. Recently, the Ate pairing, originally defined for elliptic curves, has been generalized to hyperelliptic curves [15] as well. Although the Eta and Ate pairings hold the record for speed at the present time, we will focus our attention on the Tate pairing in this paper. The main reason is that the Tate pairing is uniformly available across a wide range of hyperelliptic curves and subgroups, whereas the Eta pairing is only defined for supersingular curves and the Ate pairing incurs a huge performance penalty in the context of ordinary genus 2 curves ([15, Table 6]).

Previous work for computing pairings on hyperelliptic curves only considered using affine coordinates. Motivated by Chatterjee *et. al.*'s work [7], we address the efficient implementation of the Tate pairing on genus 2 hyperelliptic curves over large prime fields in projective coordinates in this contribution. We first derive new explicit formulae for the group operations for genus 2 hyperelliptic curves in projective and new (weighted projective) coordinates, respectively. Letting I denote a field inversion, M a field multiplication, and S a field squaring, we find in the context of pairing computations that compared to Lange's formulae [24], our mixed-addition formulae can save $5M+1S$ and $3M$ in projective and new coordinates, respectively, whereas our doubling formulae can save $2M+2S$ and $3M+2S$ in projective and new coordinates, respectively. We then show how to encapsulate the computation of the line function with the mixed addition and doubling formulae in new coordinates, and how to omit some operations which are cancelled by the final exponentiation in the encapsulated method. Our encapsulated explicit formulae can be applied to pairing computations on both supersingular and non-supersingular genus 2 hyperelliptic curves over prime fields. Finally, we describe an efficient implementation of the Tate pairing on a non-supersingular genus 2 hyperelliptic curve with an embedding degree of 2 over prime fields as a case study. To our knowledge, this is the first concrete implementation of pairing computations on non-supersingular genus 2 curves.

This paper is organized as follows. Section 2 gives an overview of the Tate pairing on hyperelliptic curves and Miller's algorithm for computing the pairing. In Section 3 we describe new explicit formulae which encapsulate group operations and line computations for genus 2 curves over prime fields. Section 4 shows how to apply various techniques from the literature to accelerate the pairing computation on a specific non-supersingular genus 2 curve over prime fields, analyzes the computational complexity of computing the Tate pairings and gives implementation results. Finally, Section 5 concludes this contribution.

2 Mathematical Background

In this section, we present a brief introduction to the definition of the Tate pairing on hyperelliptic curves and also review Miller's algorithm for computing pairings. For more details, the reader is referred to [1].

2.1 Tate Pairing on Hyperelliptic Curves

Let $\mathbb{F}_q$ be a finite field with q elements, and $\overline{\mathbb{F}}_q$ be its algebraic closure. Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$, and let $\mathcal{J}_C$ denote the degree zero divisor class group of C. We say that a subgroup of the divisor class group $\mathcal{J}_C(\mathbb{F}_q)$ has *embedding degree k* if the order n of the subgroup divides $q^k - 1$, but does not divide $q^i - 1$ for any $0 < i < k$. For our purpose, n should be a (large) prime with $n \mid \#\mathcal{J}_C(\mathbb{F}_q)$ and $\gcd(n, q) = 1$. Let $\mathcal{J}_C(\mathbb{F}_{q^k})[n]$ be the n-torsion group and $\mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k})$ be the quotient group. Then the Tate pairing is a well defined, non-degenerate, bilinear map [13]:

$$\langle \cdot, \cdot \rangle_n : \mathcal{J}_C(\mathbb{F}_{q^k})[n] \times \mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^n,$$

defined as follows: let $D_1 \in \mathcal{J}_C(\mathbb{F}_{q^k})[n]$, with $\mathrm{div}(f_{n,D_1}) = nD_1$ for some rational function $f_{n,D_1} \in \mathbb{F}_{q^k}(C)^*$. Let $D_2 \in \mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k})$ with $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$ (to ensure a non-trivial pairing value). The Tate pairing of two divisor classes $\overline{D}_1$ and $\overline{D}_2$ is then defined as

$$\langle \overline{D}_1, \overline{D}_2 \rangle_n = f_{n,D_1}(D_2) = \prod_{P \in C(\overline{\mathbb{F}}_q)} f_{n,D_1}(P)^{\mathrm{ord}_P(D_2)}.$$

Note that the Tate pairing as detailed above is only defined up to n-th powers. One can show that if the function f_{n,D_1} is properly normalized, we only need to evaluate the rational function f_{n,D_1} at the effective part of the reduced divisor D_2 in order to compute the Tate pairing [15].

In practice, the fact that the Tate pairing is only defined up to n-th power is usually undesirable, and many pairing-based protocols require a unique pairing value. Hence one defines the *reduced* pairing as

$$\langle \overline{D}_1, \overline{D}_2 \rangle_n^{(q^k-1)/n} = f_{n,D_1}(D_2)^{(q^k-1)/n} \in \mu_n \subset \mathbb{F}_{q^k}^*,$$

where $\mu_n = \{u \in \mathbb{F}_{q^k}^* \mid u^n = 1\}$ is the group of n-th roots of unity. In the rest of this paper we will refer to the extra powering required to compute the reduced pairing as the *final exponentiation.*

2.2 Miller's Algorithm

The main task involved in the computation of the Tate pairing $\langle \overline{D}_1, \overline{D}_2 \rangle_n$ is to construct a rational function f_{n,D_1} such that $\mathrm{div}(f_{n,D_1}) = nD_1$. In [28] (see also [29]), Miller described a polynomial time algorithm, known universally as Miller's algorithm, to construct the function f_{n,D_1} and compute the Weil pairing on elliptic curves. However, the algorithm can be easily adapted to compute the Tate pairing on hyperelliptic curves.

Let $G_{iD_1,jD_1} \in \mathbb{F}_{q^k}(C)^*$ be a rational function with $\mathrm{div}(G_{iD_1,jD_1}) = iD_1 + jD_1 - (iD_1 \oplus jD_1)$ where $\oplus$ is the group law on $\mathcal{J}_C$ and $(iD_1 \oplus jD_1)$ is reduced. Miller's algorithm constructs the rational function f_{n,D_1} based on the following iterative formula:

$$f_{i+j,D_1} = f_{i,D_1} f_{j,D_1} G_{iD_1,jD_1} \ .$$

Algorithm 1 shows the basic version of Miller's algorithm for computing the reduced Tate pairing on hyperelliptic curves according to the above iterative relation. Essentially, computing the Tate pairing with Miller's algorithm amounts to performing a scalar multiplication of a reduced divisor and evaluating certain intermediate rational functions which appear in the process of the divisor class addition. A more detailed version of Miller's algorithm for hyperelliptic curves can be found in [15].

Algorithm 1 Miller's Algorithm for Hyperelliptic Curves (basic version)

```
IN: $\overline{D}_1 \in \mathcal{J}_C(\mathbb{F}_{q^k})[n], \overline{D}_2 \in \mathcal{J}_C(\mathbb{F}_{q^k})$, represented by $D_1$ and $D_2$ with $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$
OUT: $\langle D_1, D_2 \rangle_n^{(q^k-1)/n}$
1. $f \leftarrow 1, T \leftarrow D_1$
2. for $i \leftarrow \lfloor \log_2(n) \rfloor - 1$ downto 0 do
3.     $\triangleright$ Compute $T'$ and $G_{T,T}(x,y)$ such that $T' = 2T - \mathrm{div}(G_{T,T})$
4.        $f \leftarrow f^2 \cdot G_{T,T}(D_2), \overline{T} \leftarrow [2]\overline{T}$
5. if $n_i = 1$ then
6.     $\triangleright$ Compute $T'$ and $G_{T,D_1}(x,y)$ such that $T' = T + D_1 - \mathrm{div}(G_{T,D_1})$
7.        $f \leftarrow f \cdot G_{T,D_1}(D_2), \overline{T} \leftarrow \overline{T} \oplus \overline{D}_1$
8. Return $f^{(q^k-1)/n}$
```

3 Encapsulated Computation on Genus 2 Hyperelliptic Curves

In this section we generalize the idea of encapsulated add-and-line and encapsulated double-and-line proposed in [7] to genus 2 hyperelliptic curves over large prime fields. Note that, in the process of computing Tate pairings, one inversion is required for each divisor class addition and doubling, and the calculation of the inversion of an element in large characteristic is usually quite expensive. Therefore, to avoid inversions, we need to derive efficient inversion-free explicit formulae for genus 2 hyperelliptic curves in the context of pairing computations.

Lange [24] presented efficient explicit formulae for the group operations on genus 2 curves using various systems of coordinates. In the projective coordinate system, the quintuple $[U_1, U_0, V_1, V_0, Z]$ corresponds to the affine class $[x^2 + U_1/Zx + U_0/Z, V_1/Zx + V_0/Z]$ in Mumford representation [31], whereas in the new coordinate system the sextuple $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ stands for the affine class $[x^2 + U_1/Z_1^2 x + U_0/Z_1^2, V_1/(Z_1^3 Z_2)x + V_0/(Z_1^3 Z_2)]$. Lange's formulae are designed to be used in the context of computing scalar multiplications, and do not explicitly calculate all of the rational functions required in Miller's algorithm. However, one can extract the rational functions required from the formulae in [24] at the cost of 3 extra field multiplications.

Choie and Lee [8] modified Lange's explicit formulae in affine coordinates to reduce the cost of extracting the rational functions required in Miller's algorithm. The formulae presented in [8] require $1I + 23M + 3S$ and $1I + 23M + 5S$ in $\mathbb{F}_p$ for divisor class addition and doubling, respectively, thereby saving 2 field multiplications over the previous method. Ó hÉigeartaigh and Scott [17] further optimized the doubling formula proposed in [8] for supersingular genus 2 curves over $\mathbb{F}_p$ of the form $y^2 = x^5 + a$ by saving 1 multiplication and 1 squaring. Unfortunately, some minor inaccuracies have arisen in the explicit formulae in [8, 17]. We have taken this opportunity to correct the inaccuracies, and the resulting divisor class addition and doubling formulae in affine coordinates (with some slight modifications for our purposes) are presented in Table 9 and Table 10 of appendix B.

Based on the explicit formulae in Table 9 and Table 10 of appendix B, we derive new explicit mixed-addition and doubling formulae in the projective and new coordinate systems in the context of pairing computations, respectively. Since the explicit formulae in new coordinates are slightly more efficient than those in projective coordinates, we use new coordinates to represent divisor classes in the main presentation. The mixed-addition and doubling formulae in projective coordinates can be found in Table 11 and Table 12 of appendix B. We will explain how to encapsulate the group operations and the line computations in the following subsections. To increase performance, we also enlarge the set of coordinates to $[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2]$ as in [24], where $z_1 = Z_1^2$ and $z_2 = Z_2^2$.

3.1 Encapsulated Mixed Divisor Addition and Line Computation

In this subsection, we show how to encapsulate the computation of the line function with the divisor class addition in new coordinates. Given two divisor classes $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1, 1, 1, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$ in new coordinates as inputs, Table 1 describes an explicit mixed-addition formula which calculates a divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $E_1 + E_2 = E_3 + \mathrm{div}\left(\frac{y - l(x)}{u_3(x)}\right)$ in the most common case.

Hence, our new explicit formula requires $36M + 5S$ for computing the divisor class addition in new coordinates. Table 2 summarizes the computational cost of calculating the divisor class addition and extracting the line function in various coordinate systems.

From Table 2 we note that in the context of pairing computations our mixed-addition formulae can save $5M + 1S$ in the projective coordinate system and $3M$ in the new coordinate system, respectively, when compared to the formulae given by Lange [24]. Therefore, our newly derived mixed-addition formulae do not introduce any additional cost for extracting the rational function required in Miller's algorithm. From this point of view, we believe that the mixed-addition formulae in Table 1 and Table 11 are optimal.

In the new coordinate system, the rational function $c(x, y) = y - l(x)$ that is required in Miller's algorithm has the following form:

$$c(x, y) = y - \left(\frac{s_1'}{r z_{23}} x^3 + \frac{l_2}{r z_{24}} x^2 + \frac{l_1}{r z_{24}} x + \frac{l_0}{r z_{24}}\right),$$

where $s_1', l_2, l_1, l_0, r, z_{23}$ and $z_{24} = z_{21} z_{23}$ are computed in Table 1. By defining the auxiliary rational function $c'(x, y) = (r z_{24}) c(x, y)$, we obtain

$$\begin{aligned} c'(x, y) &= (r z_{24}) y - ((s_1' z_{21}) x^3 + l_2 x^2 + l_1 x + l_0) \\ &= (\tilde{r} z_{21}) y - ((s_1' z_{21}) x^3 + l_2 x^2 + l_1 x + l_0), \end{aligned}$$

Table 1. Mixed-Addition Formula on a Genus 2 Curve over $\mathbb{F}_p$ (New Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1, 1, 1, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_{31}, Z_{32}, z_{31}, z_{32}] = \overline{E}_1 \oplus \overline{E}_2$ $l(x)$ such that $E_1 + E_2 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	
Step	**Expression**	**Cost**
1	**Compute resultant and precomputations:** $z_{23} = Z_{21}Z_{22}, z_{24} = z_{21}z_{23}, \tilde{U}_{11} = U_{11}z_{21}, \tilde{U}_{10} = U_{10}z_{21}, y_1 = \tilde{U}_{11} - U_{21}$ $y_2 = U_{20} - \tilde{U}_{10}, y_3 = U_{11}y_1, y_4 = y_2 + y_3, r = y_2y_4 + y_1^2U_{10}$	$7M, 1S$
2	**Compute almost inverse of u_2 mod u_1:** $inv_1 = y_1, inv_0 = y_4$	–
3	**Compute s':** $w_0 = V_{10}z_{24} - V_{20}, w_1 = V_{11}z_{24} - V_{21}, w_2 = inv_0w_0$ $w_3 = inv_1w_1, s'_1 = y_1w_0 + y_2w_1, s'_0 = w_2 - U_{10}w_3$	$7M$
4	**Precomputations:** $\tilde{r} = rz_{23}, R = \tilde{r}^2, Z_{31} = s'_1Z_{21}, Z_{32} = \tilde{r}Z_{21}, z_{31} = Z_{31}^2, z_{32} = Z_{32}^2, \tilde{s}'_0 = s'_0z_{21}$	$4M, 3S$
5	**Compute l:** $l_2 = s'_1U_{21} + \tilde{s}'_0, l_0 = s'_0U_{20} + rV_{20}, l_1 = (s'_1 + s'_0)(U_{21} + U_{20}) - s'_1U_{21} - s'_0U_{20} + rV_{21}$	$5M$
6	**Compute U_3:** $w_1 = \tilde{U}_{11} + U_{21}, U_{31} = s'_1(2\tilde{s}'_0 - s'_1y_1) - z_{32}, l'_1 = l_1s'_1$ $U_{30} = \tilde{s}'_0(s'_0 - 2s'_1U_{11}) + s_1'^2(y_3 - \tilde{U}_{10} - U_{20}) + 2l'_1 + Rw_1$	$7M, 1S$
7	**Compute V_3:** $w_1 = l_2s'_1 - U_{31}, V_{30} = U_{30}w_1 - z_{31}(l_0s'_1), V_{31} = U_{31}w_1 + z_{31}(U_{30} - l'_1)$	$6M$
Sum		$36M, 5S$

where $\tilde{r} = rz_{23}$ is also available in Table 1. Note that the result of evaluating the function $c(x, y)$ at an image divisor D_2 will be raised to the power $(q^k - 1)/n$ $(k > 1)$ in the last step of Miller's algorithm. For efficiency reasons, the first input to the Tate pairing is usually restricted to the 1-eigenspace of the Frobenius endomorphism on $\mathcal{J}_C[n]$. Therefore, we have the following relation

$$\begin{aligned} c(D_2)^{(q^k-1)/n} &= (c(D_2)^{q-1})^{(q^{k-1}+q^{k-2}+\ldots+1)/n} \\ &= ((c'(D_2)/(rz_{24}))^{q-1})^{(q^{k-1}+q^{k-2}+\ldots+1)/n} \\ &= (c'(D_2)^{q-1})^{(q^{k-1}+q^{k-2}+\ldots+1)/n} \quad (\text{since } (rz_{24})^{q-1} = 1) \\ &= c'(D_2)^{(q^k-1)/n}. \end{aligned}$$

The above relation means that in new coordinates we can work with the rational function $c'(x, y)$ instead of $c(x, y)$ without altering the value of the resulting Tate pairing. For the same reason we also work with the rational function $u'_3(x) = z_{31}x^2 + U_{31}x + U_{30}$ instead of $u_3(x) = x^2 + \frac{U_{31}}{z_{31}}x + \frac{U_{30}}{z_{31}}$ for both divisor addition and divisor doubling.

3.2 Encapsulated Divisor Doubling and Line Computation

In this subsection, we describe how to encapsulate the computation of the line function with the divisor class doubling in new coordinates. Given a divisor class $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}]$ in new coordinates as an input, Table 3 describes an explicit doubling formula which calculates a divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $2E_1 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in the most common case.

Table 2. Divisor Class Addition in Different Systems and in Odd Characteristic

Reference	Coordinate Type	Addition	Mixed Addition	Extracting Line Function $l(x)$
Miyamoto *et al.* [30]	Affine	$1I, 24M, 2S$	–	$3M$
	Projective	$54M$	–	$3M$
Lange [24]	Affine	$1I, 22M, 3S$	–	$3M$
	Projective	$47M, 4S$	$40M, 4S$	$3M$
	New	$47M, 7S$	$36M, 5S$	$3M$
Choie and Lee [8]	Affine	$1I, 23M, 3S$	–	no cost
Our work	Projective	–	$\mathbf{38M, 3S}$ **Table 11**	**no cost**
	New	–	$\mathbf{36M, 5S}$ **Table 1**	**no cost**

Therefore, our new explicit formula needs $39M + 6S$ to double a divisor class in new coordinates. Table 4 summarizes the computational cost of doubling a divisor class and extracting the line function in various coordinate systems.

From Table 4 we note that in the context of pairing computations our doubling formulae can save $2M + 2S$ and $3M + 2S$ in projective and new coordinates, respectively, when compared to the formulae given by Lange [24]. Therefore, our doubling formulae do not introduce any additional cost for extracting the rational function required in Miller's algorithm. From this point of view, we believe that the doubling formulae in Table 3 and Table 12 are also optimal.

In the new coordinate system, the rational function $c(x, y) = y - l(x)$ that is required in Miller's algorithm has the following form:

$$c(x, y) = y - \left(\frac{s_1'}{rz_{13}}x^3 + \frac{l_2}{rz_{14}}x^2 + \frac{l_1}{rz_{14}}x + \frac{l_0}{rz_{14}}\right),$$

where s_1', l_2, l_1, l_0, r and z_{13} are available in Table 3, and $z_{14} = z_{11}z_{13}$. By defining the auxiliary rational function $c'(x, y) = (rz_{14})c(x, y)$, we obtain

$$\begin{aligned} c'(x, y) &= (rz_{14})y - ((s_1'z_{11})x^3 + l_2x^2 + l_1x + l_0) \\ &= (\tilde{r}z_{11})y - ((s_1'z_{11})x^3 + l_2x^2 + l_1x + l_0), \end{aligned}$$

where $\tilde{r} = rz_{13}$ is also available in Table 3. With the same argument as the case of the mixed-addition, we have the relation $c(D_2)^{(q^k-1)/n} = c'(D_2)^{(q^k-1)/n}$ for an image divisor D_2. Therefore, we can simply work with the rational function $c'(x, y)$ instead of $c(x, y)$ without altering the value of the resulting Tate pairing in the new coordinate system.

[1] In [1] and [24], Lange describes explicit formulae for doubling a reduced divisor class, which cost $38M+6S$ and $34M + 7S$ in projective and new coordinates, respectively. Unfortunately, these formulae include some errors. For example, in Step 4 of Lange's doubling formulae (see Algorithms 14.23 and 14.26 in [1]), the computation of s_1 is wrong because the denominators of inv_0 and inv_1 are different, so are those of t_0 and t_1. We modify the Algorithms 14.23 and 14.26 of [1] (the correct explicit doubling formulae in projective and new coordinates are available upon request) and show the computational cost of the correct doubling formulae in the above Table 4.

Table 3. Doubling Formula on a Genus 2 Curve over $\mathbb{F}_p$ (New Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_{31}, Z_{32}, z_{31}, z_{32}] = [2]\overline{E}_1$ $l(x)$ such that $2E_1 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant and precomputations:** $\tilde{U}_{10} = U_{10}z_{11}, \tilde{V}_{11} = 2V_{11}, \tilde{V}_{10} = 2V_{10}, \tilde{z}_{11} = \tilde{V}_{10}z_{11}, w_0 = V_{11}^2$ $w_1 = U_{11}^2, w_2 = 4w_0, w_3 = \tilde{z}_{11} - U_{11}\tilde{V}_{11}, r = \tilde{U}_{10}w_2 + \tilde{z}_{11}w_3$	$5M, 2S$
2	**Compute almost inverse:** $inv_1' = -\tilde{V}_{11}, inv_0' = w_3$	–
3	**Compute $\boldsymbol{k'}$:** $z_{11}' = z_{11}^2, z_{11}'' = z_{11}z_{11}', w_3 = f_3z_{11}' + w_1, w_4 = 2\tilde{U}_{10}$ $k_1' = z_{12}(2w_1 + w_3 - w_4), k_0' = z_{12}(U_{11}(2w_4 - w_3) + f_2z_{11}'') - w_0$	$6M, 1S$
4	**Compute $\boldsymbol{s'}$:** $w_0 = k_0'inv_0', w_1 = k_1'inv_1', s_1' = z_{11}(\tilde{z}_{11}k_1' - \tilde{V}_{11}k_0'), s_0' = w_0 - \tilde{U}_{10}w_1$	$6M$
5	**Precomputations:** $z_{13} = Z_{11}Z_{12}, \tilde{r} = rz_{13}, R = \tilde{r}^2, Z_{31} = s_1'Z_{11}$ $Z_{32} = \tilde{r}Z_{11}, z_{31} = Z_{31}^2, z_{32} = Z_{32}^2, \tilde{s}_0' = s_0'z_{11}$	$5M, 3S$
6	**Compute $\boldsymbol{l}$:** $l_2 = s_1'U_{11} + \tilde{s}_0', l_0 = s_0'U_{10} + rV_{10}, r' = rV_{11}$ $l_1 = (s_1' + s_0')(U_{11} + U_{10}) - s_1'U_{11} - s_0'U_{10} + r'$	$5M$
7	**Compute $\boldsymbol{U_3}$:** $U_{30} = 2(r's_1' + RU_{11}) + s_0'\tilde{s}_0', U_{31} = 2s_1'\tilde{s}_0' - z_{32}$	$4M$
8	**Compute $\boldsymbol{V_3}$:** $w_1 = l_2s_1' - U_{31}, V_{30} = U_{30}w_1 - z_{31}(l_0s_1'), V_{31} = U_{31}w_1 + z_{31}(U_{30} - l_1s_1')$	$7M$
Sum		$38M, 6S$

4 Implementing the Tate Pairing

In this section, we describe various techniques to efficiently compute the Tate pairing on a non-supersingular genus 2 curve over $\mathbb{F}_p$ and give timings for our implementation.

4.1 The Non-supersingular Pairing-Friendly Genus 2 Curve

There are only a few techniques that have been proposed for constructing non-supersingular curves of genus $g \geq 2$ and low embedding degree for pairing-based cryptography — see [14, 11, 19, 22] for example. It seems to be hard to generate pairing-friendly curves in this case due to the complicated algebraic structure of hyperelliptic curves. By modeling on the Cocks-Pinch method for constructing pairing-friendly elliptic curves [9], Freeman generated the first examples of non-supersingular pairing-friendly genus 2 curves [11]. In our implementations, we will use an example from [11], which gives a genus 2 curve whose Jacobian has embedding degree 2 with respect to the prime $n = 2^{160} + 7$. The curve is given by the equation

$$C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$$

over $\mathbb{F}_p$. The hexadecimal representations of the curve coefficients f_3, f_2, f_1 and f_0, the subgroup order n, and the characteristic p of the prime field can be found in appendix A.

Table 4. Divisor Class Doubling in Different Systems and in Odd Characteristic

Reference	Coordinate Type	Doubling	Extracting Line Function $l(x)$
Miyamoto *et al.* [30]	Affine	$1I, 23M, 4S$	$3M$
	Projective	$53M$	$3M$
Lange [24]	Affine	$1I, 22M, 5S$	$3M$
	Projective	$40M, 6S^1$	$3M$
	New	$38M, 8S^1$	$3M$
Choie and Lee [8]	Affine	$1I, 23M, 5S$	no cost
Ó hÉigeartaigh and Scott [17]	Affine	$1I, 22M, 4S$	no cost
Our work	Projective	$\mathbf{41M, 4S}$ **Table 12**	**no cost**
	New	$\mathbf{38M, 6S}$ **Table 3**	**no cost**

Note that $n \mid (p+1)$ and therefore $p-1$ divides the final exponentiation $(p^2-1)/n$ of the Tate pairing. As observed in [3, 7], this implies that the final exponentiation eliminates all terms defined over $\mathbb{F}_p$ and hence one can freely manipulate the intermediate results during the computation of Tate pairing by multiplying or dividing by any element in $\mathbb{F}_p^*$.

Let $c \in \mathbb{F}_p$ be a quadratic non-residue over $\mathbb{F}_p$. A quadratic twist of C, denoted by C_t, over $\mathbb{F}_p$ is defined by the following equation

$$C_t : y^2 = x^5 + c^2 f_3 x^3 + c^3 f_2 x^2 + c^4 f_1 x + c^5 f_0.$$

Let $\mathcal{J}_{C_t}(\mathbb{F}_{p^2})$ be the Jacobian of C_t when considering C_t as a curve defined over $\mathbb{F}_{p^2}$, and $\overline{D}_t = [u_t, v_t]$ be an element of $\mathcal{J}_{C_t}(\mathbb{F}_{p^2})$ in Mumford representation. It is known that $C_t(\mathbb{F}_{p^2})$ is isomorphic to $C(\mathbb{F}_{p^2})$, whereas the isomorphic relation does not hold for $C_t(\mathbb{F}_p)$ and $C(\mathbb{F}_p)$ [23]. Therefore, we can construct the following isomorphism of Jacobians

$$\begin{array}{rcl} \psi : \mathcal{J}_{C_t}(\mathbb{F}_{p^2}) & \longrightarrow & \mathcal{J}_C(\mathbb{F}_{p^2}) \\ \overline{D}_t & \longmapsto & \overline{D} \\ [u_t, v_t] & \longmapsto & [u, v] \end{array}$$

by applying the isomorphism

$$\begin{array}{rcl} \phi : C_t(\mathbb{F}_{p^2}) & \longrightarrow & C(\mathbb{F}_{p^2}) \\ (x_t, y_t) & \longmapsto & (x, y) \\ (x_t, y_t) & \longmapsto & (c^{-1}x_t, c^{-5/2}y_t) \end{array}$$

to each point $P = (x_t, y_t)$ in the support of the divisor D_t.

4.2 Finite Field Arithmetic

In our implementation, the curve C has embedding degree $k = 2$. Therefore we first need to construct the quadratic extension field $\mathbb{F}_{p^2}$. Since the prime p in this paper is congruent to 5 modulo 12, the quadratic extension field $\mathbb{F}_{p^2}$ can be constructed by the irreducible binomial $x^2 + 3$. Letting β denote -3, the elements of the field $\mathbb{F}_{p^2}$ can be represented as $a + b\sqrt{\beta}$, where $a, b \in \mathbb{F}_p$. By using the Karatsuba multiplication technique [21], a multiplication of two elements in $\mathbb{F}_{p^2}$ costs 3 multiplications in $\mathbb{F}_p$.

4.3 Using Degenerate Divisors and Denominator Elimination

Degenerate divisors have been widely used in the literature to speed up pairing computations on supersingular hyperelliptic curves [2, 10, 17]. Frey and Lange [12] have shown that the value of the Tate pairing is non-trivial if one restricts the second input to the embedding of $C(\mathbb{F}_{q^k})$ into $\mathcal{J}_C(\mathbb{F}_{q^k})$. In particular, when the embedding degree k is even, we can use a degenerate divisor class $\overline{P - P_\infty} \in \mathcal{J}_C(\mathbb{F}_{q^k})$ as the second argument of the Tate pairing where the coordinates of $P = (x, y) \in C(\mathbb{F}_{q^k})$ satisfy $x \in \mathbb{F}_{q^{k/2}}$ but $y \notin \mathbb{F}_{q^{k/2}}$. Note that the point P is actually on the quadratic twist of $C/\mathbb{F}_{q^{k/2}}$. Therefore, in our implementation we first generate a degenerate divisor class $\overline{D}_t = [x - x_t, y_t] \in \mathcal{J}_{C_t}(\mathbb{F}_p)$ on the twisted curve $C_t/\mathbb{F}_p$. We then use the isomorphism ψ given above to obtain the degenerate divisor class $\overline{D} = \psi(\overline{D}_t) = [x - c^{-1}x_t, c^{-5/2}y_t] \in \mathcal{J}_C(\mathbb{F}_{p^2})$ on the curve C defined over $\mathbb{F}_{p^2}$. Hence, $\overline{D}$ can be used as the second argument to Miller's algorithm. Note that the first part of $\overline{D}$, namely $x - c^{-1}x_t$, is defined over $\mathbb{F}_p$, and thus the denominator elimination technique [3] applies in this case.

4.4 Evaluating Line Functions

At each iteration of the loop, we extract the rational functions $y - l(x)$ and $u_3(x)$ from the group operations and evaluate these functions at the second argument D_2. When new coordinates are used, we can work with $c'(x, y) = (\tilde{r}z_{21})y - ((s_1'z_{21})x^3 + l_2x^2 + l_1x + l_0)$ and $u_3'(x) = z_{31}x^2 + U_{31}x + U_{30}$ for group addition, and $c'(x, y) = (\tilde{r}z_{11})y - ((s_1'z_{11})x^3 + l_2x^2 + l_1x + l_0)$ and $u_3'(x) = z_{31}x^2 + U_{31}x + U_{30}$ for group doubling as described in Section 3, where $\tilde{r}, z_{11}, z_{21}, z_{31}, s_1', l_2, l_1$ and l_0 are from Table 1 and Table 3. We consider the following two cases for evaluating line functions at the divisor D_2:

1. D_2 is a degenerate divisor generated by the method in Section 4.3. In this case, D_2 can be represented by $[x - x_2, y_2] \in \mathcal{J}_C(\mathbb{F}_{p^2})$ for which $x_2 \in \mathbb{F}_p$ and $y_2 \notin \mathbb{F}_p$. Since x_2^2 and x_2^3 can be precomputed, this leaves $6M$ over $\mathbb{F}_p$ to be computed each time the function $c'(x, y)$ is evaluated. In particular, denominator elimination is applicable in this case and we do not need to evaluate the function $u_3(x)$ at D_2. Therefore, the total cost of evaluating the rational functions at a degenerate divisor D_2 is given as $6M$ in $\mathbb{F}_p$ per iteration of the loop, with a precomputation of $1M + 1S$ in $\mathbb{F}_p$.
2. D_2 is a general divisor in Mumford representation, namely $D_2 = [u_2(x), v_2(x)] = [x^2 + u_{21}x + u_{20}, v_{21}x + v_{20}] \in \mathcal{J}_C(\mathbb{F}_{p^2})$. Note that the Mumford representation of divisors essentially gives the symmetric functions of the coordinates of the points in the support of the divisor. Therefore, we can use these symmetric functions to obtain an explicit formula for evaluating the rational functions $c'(x, y)$ and $u_3'(x)$ at D_2, which only uses the coefficients of $u_2(x)$ and $v_2(x)$. Table 5 describes the efficient explicit formula for computing $c'(D_2)$ and $u_3'(D_2)$.

   Note that, in many of the multiplications in Table 5, one of the operands is in $\mathbb{F}_p$. Hence, a multiplication in $\mathbb{F}_{p^2}$ only needs $2M$ in $\mathbb{F}_p$ in this case. The total cost of evaluating the rational functions at a general divisor D_2 is given as $51M + 3S$ in $\mathbb{F}_p$ per iteration of the loop, with a precomputation of $13M + 3S$ in $\mathbb{F}_{p^2}$.

Table 5. Evaluating $c'(x,y)$ and $u'_3(x)$ at a General Divisor D_2 in New Coordinates

Input	$c'(x,y) = (\tilde{r}z_{21})y - ((s'_1 z_{21})x^3 + l_2x^2 + l_1x + l_0) \in \mathbb{F}_p[x,y]$ $u'_3(x) = z_{31}x^2 + U_{31}x + U_{30} \in \mathbb{F}_p[x]$ $D_2 = [x^2 + u_{21}x + u_{20}, v_{21}x + v_{20}] \in \mathcal{J}_C(\mathbb{F}_{p^2})$	
Output	$c'(D_2), u'_3(D_2) \in \mathbb{F}_{p^2}$	

Precomputations	Cost
$t_1 = u_{20}v_{21}, t_2 = u_{21}v_{20}, t_3 = t_1 - t_2, t_4 = v_{21}t_3, t_5 = v_{20}^2, t_6 = t_4 + t_5, t_7 = u_{21}v_{21}$ $t_8 = 2v_{20} - t_7, t_9 = t_1 + t_3, t_{10} = u_{21}t_3, t_{11} = u_{20}v_{20}, t_{12} = t_{10} + 2t_{11}, t_{13} = u_{21}^2$ $t_{14} = t_3t_{13}, t_{15} = 2t_3 - t_2, t_{16} = u_{20}t_{15}, t_{17} = t_{14} - t_{16}, t_{18} = u_{20}u_{21}, t_{19} = u_{20}^2$ $t_{20} = t_{19}u_{20}, t_{21} = t_{19}u_{21}, t_{22} = t_{13} - 2u_{20}, t_{23} = u_{20}t_{22}, t_{24} = t_{22} - u_{20}, t_{25} = u_{21}t_{24}$	$13M, 3S$ in $\mathbb{F}_{p^2}$

Computing $c'(D_2)$	Cost
$w_1 = \tilde{r}z_{21}, w_2 = s'_1z_{21}, w_3 = w_1t_6, w_4 = w_2t_{17}, w_5 = l_2t_{12}, w_6 = l_1t_9, w_7 = l_0t_8$ $w_8 = w_3 - w_4 + w_5 - w_6 - w_7, w_9 = w_1w_8, w_{10} = w_2t_{20}, w_{11} = l_2t_{21}, w_{12} = l_1t_{23}$ $w_{13} = l_0t_{25}, w_{14} = w_{10} - w_{11} + w_{12} - w_{13}, w_{15} = w_2w_{14}, w_{16} = l_2t_{19}, w_{17} = l_1t_{18}$ $w_{18} = l_0t_{22}, w_{19} = w_{16} - w_{17} + w_{18}, w_{20} = l_2w_{19}, w_{21} = l_1u_{20}, w_{22} = l_0u_{21}$ $w_{23} = w_{21} - w_{22}, w_{24} = l_1w_{23}, w_{25} = l_0^2, c'(D_2) = w_9 + w_{15} + w_{20} + w_{24} + w_{25}$	$38M, 1S$ in $\mathbb{F}_p$

Computing $u'_3(D_2)$	Cost
$h_1 = z_{31}^2, h_2 = z_{31}U_{30}, h_3 = U_{30}^2, h_4 = h_1t_{19}, h_5 = U_{31}u_{20}, h_6 = z_{31}t_{18}, h_7 = U_{30}u_{21}$ $h_8 = h_5 - h_6 - h_7, h_9 = U_{31}h_8, h_{10} = h_2t_{22}, u'_3(D_2) = h_3 + h_4 + h_9 + h_{10}$	$13M, 2S$ in $\mathbb{F}_p$

4.5 Final Exponentiation

For a genus 2 curve with an embedding degree of $k = 2$, the output of Miller's algorithm must be exponentiated to the power of $(p^2 - 1)/n$. The final exponentiation can be expressed in terms of operations in the base field $\mathbb{F}_p$. Letting $f = a + b\sqrt{\beta} \in \mathbb{F}_{p^2}$ denote the output of Miller's algorithm, we can compute the final exponentiation as follows:

$$f^{\frac{p^2-1}{n}} = \left(\frac{a - b\sqrt{\beta}}{a + b\sqrt{\beta}}\right)^{\frac{p+1}{n}} = \left(\frac{a^2 - 3b^2 + \sqrt{\beta}\left((a-b)^2 - (a^2+b^2)\right)}{a^2 + 3b^2}\right)^{\frac{p+1}{n}} .$$

We first calculate the expression in the parenthesis with $1I + 2M + 3S$ in $\mathbb{F}_p$, followed by an expensive exponentiation by $(p+1)/n$ which is executed in the arithmetic in $\mathbb{F}_{p^2}$.

4.6 Efficiency Comparison and Analysis

Since our encapsulated explicit formulae are applicable to pairing computations on both supersingular and non-supersingular genus 2 curves, we first show how our method can be used to improve previous implementations on supersingular genus 2 curves. We then analyze the case of non-supersingular genus 2 curves.

In [8] and [17], the authors considered the pairing computation on a family of supersingular genus 2 hyperelliptic curves with embedding degree 4 in affine coordinates. The curves are defined by the equation $y^2 = x^5 + a$, where $a \in \mathbb{F}_p^*$ and $p \equiv 2, 3 \bmod 5$. Note that our explicit doubling formulae only need $39M + 4S$ and $35M + 5S$ in projective and new coordinates, respectively, for this family of curves since the curve coefficients f_2 and f_3 are zero. Assume that the order n of the subgroup is about 160 bits. Following the same analysis as in [8] and [17], we compare the theoretical cost of computing the Tate pairing on this family of curves in different coordinate systems (without including the cost of the final exponentiation) in Table 6.

Table 6. Theoretical Complexity of Miller's Algorithm in Different Systems

Reference	Coordinate Type	Subgroup Order	Cost
Choie and Lee [8]	Affine	Random	$240I, 17688M, 2163S$
Ó hÉigeartaigh and Scott [17]	Affine	Solinas Prime	$162I, 10375M, 645S$
Our work	Projective	Random	$20337M, 881S$
		Solinas Prime	$13449M, 647S$
	New	Random	$19777M, 1201S$
		Solinas Prime	$12967M, 811S$

We assume that field squarings have cost $S = 0.8M$. Then our encapsulated method is faster than that of [8] whenever $I/M > 5.5$. Moreover, our algorithm can achieve better performance than that of [17] whenever $I/M > 16.82$. These conditions usually hold for large prime field arithmetic on modern processors [27]. Therefore, for sufficiently large I/M, our method based on the encapsulated explicit formulae will be superior. The actual performance improvements by using our algorithm depend on the specific implementation and the processor architecture.

Next, we analyze the computational complexity of computing the Tate pairing using the non-supersingular genus 2 curve with embedding degree 2 (see Section 4.1). Note that the group order $n = 2^{160} + 7 = 2^{160} + 2^3 - 1$ is a Solinas prime [35]. The encapsulated explicit formulae for performing the group operations and extracting the rational functions in new coordinates are used (See Section 3). Furthermore, the degenerate divisor generated by using the technique in Section 4.3 is used as the second argument to Miller's algorithm. Based on these optimizations, the theoretical cost for computing the Tate pairing is given as (again without including the cost of the final exponentiation)

$$(\log_2 n)(T_D + T_c + T_{sk} + T_{mk}) + 2(T_A + T_c + T_{mk}),$$

where $T_A = 36M + 5S$ is the cost of adding two general divisors and extracting the rational functions with the formula in Table 1, $T_D = 38M + 6S$ is the cost of doubling a general divisor and extracting the rational functions with the formula in Table 3, $T_c = 6M$ (with a precomputation of $1M + 1S$) is cost of evaluating the rational function $c'(x, y)$ at the degenerate divisor D_2, and $T_{sk} = 2M$ and $T_{mk} = 3M$ are respectively the cost of squaring and multiplication in $\mathbb{F}_{p^2}$. Hence, the total cost of computing the Tate pairing with our optimizations is given as $7980M + 977S$ in $\mathbb{F}_p$, whereas $163I + 4887M + 492S$ are required when using affine coordinates.

4.7 Experimental Results

In this section, experimental results are given for computing the Tate pairing using the techniques detailed in this paper for the non-supersingular genus 2 curve defined over $\mathbb{F}_p$ with embedding degree 2. All experiments were conducted on a Core 2 Duo$^{\text{TM}}$processor with a clock frequency of 2.67 GHz. The code was written in C and complied and debugged using *Microsoft Developer Studio 6*. The implementation of $\mathbb{F}_p$-arithmetic is based on various efficient algorithms in [16], where p is a 651 bits prime. Table 7 shows the timings of our finite field library and the corresponding MI-ratio. From Table 7, we note that the MI-ratio is 45.8 and $S = 0.89M$ in the target processor. Therefore, using our encapsulated explicit formulae can obtain a 30.8% performance improvement over working with affine coordinates theoretically.

Table 7. Timings of Prime Field $\mathbb{F}_p$ Library

# of bits of p	Multiplication (M)	Squaring (S)	Inversion (I)	MI-ratio
651	$4.59\mu s$	$4.08\mu s$	$210\mu s$	45.8

Table 8 gives experimental results for the implementation of the Tate pairing for the (160/1024) security level. All of the timings are given in milliseconds and three cases are included in the Table. The first case is the time taken to compute the Tate pairing when a degenerate devisor is used as the second argument to Miller's algorithm and the denominator technique is applied. The second case gives the time when a general divisor with Mumford representation is used as the second input to the algorithm. The third case is the timing for computing the Tate pairing on non-supersingular elliptic curves with the embedding degree 2 over $\mathbb{F}_p$ given by Scott [34].

Table 8. Experimental Results – (160/1024) Security Level

Case	Description	Running Time (ms)
1	Evaluating at a degenerate divisor (denominator elimination)	47.7
2	Evaluating using Mumford Representation	86.5
3	Elliptic curve pairing ($k = 2$ and $\log_2(p) \approx 512$) [34]	8.9

Note that the implementations in [17] and [34] use special assembly language routines in MIRACL [33] for field operations. Therefore, it is hard to compare our implementation with those in [17, 34]. However, the timings in Table 8 indicate that the Tate pairing on the non-supersingular genus 2 curve over $\mathbb{F}_p$ is a valid candidate for practical applications. Furthermore, the elliptic curve pairings are faster than those in the genus 2 case for a (160/1024) bit security level, due to the more complicated group operations on genus 2 curves.

5 Conclusion

In this paper, we have described how to efficiently implement pairing computations on genus 2 hyperelliptic curves over prime fields in projective coordinates. We generalize Chatterjee *et. al.*'s idea of encapsulated double-and-line computation and add-and-line computation to genus 2 curves in projective and new coordinates, respectively. We also show that some of the operations in the encapsulated method do not need to be computed since they are eliminated by the final exponentiation. Our new explicit formulae are applicable to pairing computations on both supersingular and non-supersingular genus 2 curves. Theoretical analysis shows that for pairing computations on supersingular genus 2 curves with embedding degree 4 over prime fields, our encapsulated method is faster than previously best known algorithms whenever $I/M > 16.82$. Furthermore, we also report the first efficient implementation of pairing computations on a non-supersingular genus 2 curve with embedding degree 2 over prime fields using the encapsulated explicit formulae and various known optimization techniques.

References

1. R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 2006.
2. P.L.S.M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," *Design, Codes and Cryptography*, 42:239-271, 2007.
3. P.L.S.M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient Algorithm for Pairing-Based Cryptosystems," *Advance in Cryptology - CRYPTO'2002*, ser. LNCS 2442, M. Yung Ed., Berlin, Germany: Springer-Verlag, pp. 354-368, 2002.
4. P.L.S.M. Barreto, B. Lynn, and M. Scott, "On the Selection of Pairing-Friendly Groups," *Selected Areas in Cryptography - SAC'2003*, ser. LNCS 3006, M. Matsui and R. Zuccherato Eds., Berlin, Germany: Springer-Verlag, pp. 17-25, 2003.
5. D. Boneh, and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM Journal of Computing*, 32(3):586-615, 2003.
6. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Advance in Cryptology - ASIACRYPT'2001*, ser. LNCS 2248, C. Boyd Ed., Berlin, Germany: Springer-Verlag, pp. 514-532, 2001.
7. S. Chatterjee, P. Sarkar, and R. Barua "Efficient Computation of Tate Pairing in Projective Coordinate over General Characteristic Fields," *Information Security and Cryptology - ICISC 2004*, ser. LNCS 3506, C. Park and S. Chee Eds., Berlin, Germany: Springer-Verlag, pp. 168-181, 2005.
8. Y. Choie, and E. Lee, "Implementation of Tate Pairing on Hyperelliptic Curve of Genus 2," *Information Security and Cryptology - ICISC'2003*, ser. LNCS 2971, J. I. Lim and D. H. Lee Eds., Berlin, Germany: Springer-Verlag, pp. 97-111, 2004.
9. C. Cocks, and R. G. E. Pinch "Identity-based Cryptosystems Based on the Weil Pairing," Unpublished manuscript, 2001.
10. I. Duursma, and H. S. Lee, "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$," *Advances in Cryptology - ASIACRYPT'2003*, ser. LNCS 2894, C. S. Laih, Ed. Berlin, Germany: Springer-Verlag, pp. 111-123, 2003.
11. D. Freeman, "Constructing Pairing-Friendly Genus 2 Curves over Prime Fields with Ordinary Jacobians," *Pairing-Based Cryptography - Pairing 2007*, ser. LNCS 4575, T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto, Eds. Berlin, Germany: Springer-Verlag, pp. 152-176, 2007.
12. G. Frey, and T. Lange "Fast Bilinear Maps from The Tate-Lichtenbaum Pairing on Hyperelliptic Curves," *Algorithmic Number Theory Symposium - ANTS VII*, ser. LNCS 4076, F. Hess, S. Pauli, M. Pohst, Eds. Berlin, Germany: Springer-Verlag, pp. 466-479, 2006.
13. G. Frey, and H.-G. Rück, "A Remark Concerning m-Divisibility and the Discrete Logarithm Problem in the Divisor Class Group of Curves," *Mathematics of Computation*, 62(206):865-874, 1994.
14. S. D. Galbraith, J. F. McKee, and P. C. Valença "Ordinary Abelian Varieties Having Small Embedding Degree," *Finite Fields and Their Applications*, vol. 13, iss. 4, pp. 800-814, 2007.
15. R. Granger, F. Hess, R. Oyono, N. Thériault, and F. Vercauteren, "Ate Pairing on Hyperelliptic Curves," *Advance in Cryptology - EUROCRYPT'2007*, ser. LNCS 4515, M. Naor, Ed. Berlin, Germany: Springer-Verlag, pp. 430-447, 2007.
16. D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, New York, USA: Springer-Verlag, 2004.
17. C. Ó. hÉigeartaigh, and M. Scott, "Pairing Calculation on Supersingular Genus 2 Curves", *Selected Areas in Cryptography - SAC 2006*, ser. LNCS 4356, E. Biham, Amr M.Youssef, Eds. Berlin, Germany: Springer-Verlag, pp. 302-316, 2007.
18. F. Hess, N. P. Smart, and F. Vercauteren, "The Eta Pairing Revisited," in *IEEE Transactions on Information Theory*, 52(10):4595-4602, 2006.
19. L. Hitt, "Families of Genus 2 Curves with Small Embedding Degree," Cryptology ePrint Archive, Report 2007/001, 2007, `http://eprint.iacr.org/2007/001`.

20. A. Joux, “A One-Round Protocol for Tripartite Diffie-Hellman,” *Algorithmic Number Theory Symposium - ANTS IV*, ser. LNCS 1838, W. Bosma Ed., Berlin, Germany: Springer-Verlag, pp. 385-394, 2000.
21. A. Karatsuba, and Y. Ofman, ”Multiplication of Multidigit Numbers on Automata,” in *Soviet Physics Doklady. (English Translation)*, vol. 7, no. 7, pp. 595-596, 1963.
22. M. Kawazoe, and T. Takahashi, “Pairing-friendly Hyperelliptic Curves of Type $y^2 = x^5 + ax$,” Cryptology ePrint Archive, Report 2008/026, 2008, `http://eprint.iacr.org/2008/026`.
23. S. Kozaki, K. Matsuo, and Y. Shimbara, “Skew-Frobenius Maps on Hyperelliptic Curves,” *The 2007 Symposium on Cryptography and Information Security - SCIS 2007*, IEICE Japan, 1D2-4, January 2007.
24. T. Lange, “Formulae for Arithmetic on Genus 2 Hyperelliptic Curves,” in *Applicable Algebra in Engineering, Communication and Computing*, vol.15, No.5, pp. 295-328, 2005.
25. E. Lee, H.-S. Lee, and Y. Lee, “Eta Pairing Computation on General Divisors over Hyperelliptic Curves $y^2 = x^7 - x \pm 1$,” *Pairing-Based Cryptography - Pairing 2007*, ser. LNCS 4575, T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto, Eds. Berlin, Germany: Springer-Verlag, pp. 349-366, 2007.
26. A. Menezes, T. Okamoto, and S. A. Vanstone, “Reducing Elliptic Curve Logarithms to a Finite Field,” in *IEEE Transactions on Information Theory*, 39(5):1639-1646, 1993.
27. A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 1997.
28. V. S. Miller, “Short Programs for Functions on Curves,” Unpublished manuscript, 1986, available at `http://crypto.stanford.edu/miller/miller.pdf`.
29. V. S. Miller, “The Weil Pairing and Its Efficient Calculation, *Journal of Cryptology*, vol. 17, no. 4, pp. 235-261, 2004.
30. Y. Miyamoto, H. Doi, K. Matsuo, J. Chao and S. Tsujii, “A Fast Addition Algorithm of Genus Two Hyperelliptic Curve,” *The 2002 Symposium on Cryptography and Information Security - SCIS 2002*, IEICE Japan, pp. 497-502, 2002, in Japanese.
31. D. Mumford, “Tata Lectures on Theta II,” *Prog. Math.*, vol. 43. Birkhäuser, 1984.
32. R. Sakai, K. Ohgishi, and M. Kasahara, “Cryptosystems Based on Pairings,” *Proceedings of the 2000 Symposium on Cryptography and Information Security - SCIS’2002*, Okinawa, Japan, pp. 26-28, 2000.
33. M. Scott, MIRACL (Multiprecision Integer and Rational Arithmetic C/C++ Library), available at `http://www.shamus.ie/`.
34. M. Scott, “Scaling Security in Pairing-based Protocols,” Cryptology ePrint Archive, Report 2005/139, 2005, `http://eprint.iacr.org/2005/139`.
35. J. Solinas, “Generalized Mersenne Primes,” *Centre for Applied Cryptographic Research (CACR) Technical Reports*, CORR 99-39, available at `http://www.cacr.math.uwaterloo.ca/techreprots/1999/corr99-39.pdf`.

Appendix A: Curve Parameters

Here, we give the parameters of a genus 2 pairing-friendly hyperelliptic curve and its quadratic twist that are used in our implementations. The particular curve whose Jacobian has embedding degree 2 is defined by the equation

$$C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$$

with the following parameters [11] (in hexadecimal format):

– Subgroup and Prime Field Parameters:

$$\begin{aligned} n &= \texttt{00000001 00000000 00000000 00000000 00000000 00000007} \\ p &= \texttt{000006ce 80bf9f2d 63bb81f2 3fe0d77a c91f27b4 e98232c3 c42cd1e6} \\ &\ \texttt{d1674062 4579fd66 dfbf8c73 7f3984c9 5231332c 0ab42b75 1bea9cca} \\ &\ \texttt{c931cfd5 432d6f85 33827ed6 d664f8f8 0d517e5e 2571001e 596dcfa1} \end{aligned}$$

- Curve Parameters:

$f_3 =$
```
000006b1 05d3dfd6 1b4155ba e7d9117f fecaf167 572e6dd4 841b7a3e
b757fb50 48bd9d05 c780e420 d5a8b9d8 e73c243c e7b36b0a 1c3bd674
a474b9cb 0074fe3f 714ae96f 8005f1cb f88b3347 e04a67f3 ab678c11
```
$f_2 =$
```
0000067d d874a209 56fc8017 1b3ab372 d374557c 281ba89b 790a2da8
22f9adc8 47fc7d85 8c74b5e2 07ceca9d 6db2faa3 ff40c67d 9e78c8a8
bbc883d0 ff36b9c9 93c53747 a6a24a7b 30cbe7fd c9816b39 d5f9adf2
```
$f_1 =$
```
00000344 da38d95d ca5650e5 7d050a4b 043ca574 a47e6792 f09b2293
b0de1397 1cd91a1e e88ac69a 865c6666 356a500c 09b296e2 29f9a12b
7d680899 cdebbe13 701232ca 0f49ac6e 3cf7eca5 1de12bf1 7312af42
```
$f_0 =$
```
000003de 36d0749f 0083196d 40f33114 963f6662 3f606625 26473400
de9548b3 955aeba9 95cb0f33 285936b3 4e0ddad0 af078762 2c4f5f85
bc4c94ed 433c3484 a88c4976 06b85cac 59a0fa31 0c11a427 5fb371a2
```

A quadratic twist C_t of the genus 2 curve C over $\mathbb{F}_p$ is defined as

$$C_t : y^2 = x^5 + f'_3x^3 + f'_2x^2 + f'_1x + f'_0$$

with the following parameters (in hexadecimal format):

$f'_3 =$
```
000005c5 2e75e51b d76ff400 279ae1a9 ac293efa c490465a 8390bcfd
e6ddd2c0 62da99fd 058ba18b 89226255 8f93acc3 cfad67b2 1ec5a3c3
7e8c0978 eab17411 5f8e3e34 cd0db86b 5258da95 b715a69e 3b356f91
```
$f'_2 =$
```
000001b3 3f29139f f469b02a 9da4f55c 1fe50447 7c4e5f7c 287a80b5
942735db 76c27f5c e92514e4 19081dd7 c71ec32d 2a777aa5 1f16c2cc
a0e9349d e7d7bb44 a5740d43 33236e33 34c55dcb 8cd3b3fb 83d1bdd4
```
$f'_1 =$
```
00000621 ef8b25f0 377a4ea7 13384582 7c90750d 62ab3c6f 026ec877
e0f0a439 d095a682 5d79fdbf a0b2b074 b355bb45 7ac34a2d 2328b8a8
d085df02 2ed89660 d0633e0b 03529a0f 50591e44 e578e2e4 229ca1fc
```
$f'_0 =$
```
00000631 e02cbab7 aa5e6bd4 083d6920 95bc5ff8 a1309b08 2ebf9b80
6b5bf2e4 f3eee4e2 4d41d526 c78c2d1c 899014d3 abae1666 190b7629
815a7b94 a785b366 fab1239e 07d33716 2772208f df9c3f1a b74adfa5
```

Appendix B: Explicit Formulae for Genus 2 Curves over $\mathbb{F}_p$

In this appendix, we give efficient explicit formulae for group operations on genus 2 curves over $\mathbb{F}_p$ in the context of pairing computations. Table 9 and Table 10 describe how to perform group operations via explicit formulae and extract the rational functions that are required by Miller's algorithm in the affine coordinate system, whereas Table 11 and Table 12 address the cases of projective coordinates. Given two divisor classes $\overline{E}_1$ and $\overline{E}_2$, Table 9 and Table 11 compute the divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $E_1 + E_2 = E_3 + \mathrm{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in the affine and the projective coordinate systems, respectively. In the affine coordinate system, $l(x) = s_1x^3+l_2x^2+l_1x+l_0$ (see Table 9), whereas $l(x) = \frac{s_1'}{r}x^3 + \frac{l_2}{rZ_2}x^2 + \frac{l_1}{rZ_2}x + \frac{l_0}{rZ_2}$ in projective coordinates (see Table 11). For doubling a reduced divisor class $\overline{D}_1$, Table 10 and Table 12 calculate the divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $2E_1 = E_3 + \mathrm{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in affine and projective coordinates, respectively. In the affine coordinate system, $l(x) = s_1x^3+l_2x^2+l_1x+l_0$ (see Table 10), whereas $l(x) = \frac{s_1'}{r}x^3+\frac{l_2}{rZ_1}x^2+\frac{l_1}{rZ_1}x+\frac{l_0}{rZ_1}$ in projective coordinates (see Table 12).

Table 9. Addition Formula on a Genus 2 Curve over $\mathbb{F}_p$ (Affine Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [u_1, v_1]$ with $u_1 = x^2 + u_{11}x + u_{10}$ and $v_1 = v_{11}x + v_{10}$ $\overline{E}_2 = [u_2, v_2]$ with $u_2 = x^2 + u_{21}x + u_{20}$ and $v_2 = v_{21}x + v_{20}$	
Output	$\overline{E}_3 = [u_3, v_3]$ with $u_3 = x^2 + u_{31}x + u_{30}$ and $v_3 = v_{31}x + v_{30}$ $l(x)$ such that $E_1 + E_2 = E_3 + \mathrm{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant $r = \mathrm{Res}(u_1, u_2)$:** $z_1 = u_{11} - u_{21}, z_2 = u_{20} - u_{10}, z_3 = u_{11}z_1$ $z_4 = z_2 + z_3, r = z_2z_4 + z_1^2u_{10}$	$3M, 1S$
2	**Compute almost inverse $inv = (r/u_2) \bmod u_1$:** $inv_1 = z_1, inv_0 = z_4$	–
3	**Compute $s' = rs = ((v_2 - v_1)inv) \bmod u_1$:** $w_0 = v_{10} - v_{20}, w_1 = v_{11} - v_{21}, w_2 = inv_0w_0, w_3 = inv_1w_1$ $s_1' = z_1w_0 + z_2w_1, s_0' = w_2 - u_{10}w_3$ **If** $s_1' = 0$ **then goto step** 4'	$5M$
4	**Compute $s = s_1x + s_0$ and s_1^{-1}:** $w_1 = (rs_1')^{-1}, w_2 = s_1'w_1, w_3 = r^2w_1, s_1 = s_1'w_2, s_0 = s_0'w_2$	$1I, 5M, 1S$
5	**Compute $l(x) = su_2 + v_2 = s_1x^3 + l_2x^2 + l_1x + l_0$:** $l_2 = s_1u_{21} + s_0, l_0 = s_0u_{20} + v_{20}$ $l_1 = (s_1 + s_0)(u_{21} + u_{20}) - s_1u_{21} - s_0u_{20} + v_{21}$	$3M$
6	**Compute $u_3 = \mathrm{monic}\left(\frac{f-l^2}{u_1u_2}\right) = x^2 + u_{31}x + u_{30}$:** $s_0'' = w_3s_0, w_1 = u_{11} + u_{21}, u_{31} = 2s_0'' - z_1 - w_3^2$ $u_{30} = s_0''(s_0'' - 2u_{11}) + z_3 - u_{10} - u_{20} + w_3(2l_1 + w_1w_3)$	$4M, 1S$
7	**Compute $v_3 = -l \bmod u_3 = v_{31}x + v_{30}$:** $w_1 = u_{31}s_1, w_2 = l_2 - w_1, w_3 = u_{30}w_2, v_{30} = w_3 - l_0$ $v_{31} = (u_{31} + u_{30})(w_2 + s_1) - w_1 - w_3 - l_1$	$3M$
Sum		$1I, 23M, 3S$
4'	**Compute $l(x) = su_2 + v_2$:** $inv = 1/r, s_0 = s_0'inv, l_1 = s_0u_{21} + v_{21}, l_0 = s_0u_{20} + v_{20}$	$1I, 3M$
5'	**Compute $u_3 = \mathrm{monic}\left(\frac{f-l^2}{u_1u_2}\right) = x + u_{30}$:** $u_{30} = -(u_{11} + u_{21} + s_0^2)$	$1S$
6'	**Compute $v_3 = -l \bmod u_3 = v_{30}$:** $v_{30} = u_{30}(l_1 - u_{30}s_0) - l_0$	$2M$
Sum		$1I, 13M, 2S$

Table 10. Doubling Formula on a Genus 2 Curve over $\mathbb{F}_p$ (Affine Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [u_1, v_1]$ with $u_1 = x^2 + u_{11}x + u_{10}$ and $v_1 = v_{11}x + v_{10}$	
Output	$\overline{E}_3 = [u_3, v_3]$ with $u_3 = x^2 + u_{31}x + u_{30}$ and $v_3 = v_{31}x + v_{30}$ $l(x)$ such that $2E_1 = E_3 + \mathrm{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute $\boldsymbol{\tilde{v}_1 = (2v_1) \bmod u_1 = \tilde{v}_{11}x + \tilde{v}_{10}}$:** $\tilde{v}_{11} = 2v_{11}, \tilde{v}_{10} = 2v_{10}$	–
2	**Compute resultant $\boldsymbol{r = \mathrm{Res}(u_1, \tilde{v}_1)}$:** $w_0 = v_{11}^2, w_1 = u_{11}^2, w_2 = 4w_0, w_3 = u_{11}\tilde{v}_{11}$ $w_4 = \tilde{v}_{10} - w_3, r = u_{10}w_2 + \tilde{v}_{10}w_4$	$3M, 2S$
3	**Compute almost inverse $\boldsymbol{inv' = (r/(2v_1)) \bmod u_1}$:** $inv_1' = -\tilde{v}_{11}, inv_0' = w_4$	–
4	**Compute $\boldsymbol{k' = \left(\frac{f-v_1^2}{u_1}\right) \bmod u_1 = k_1'x + k_0'}$:** $w_3 = f_3 + w_1, w_4 = 2u_{10}, k_1' = 2w_1 + w_3 - w_4$ $k_0' = u_{11}(2w_4 - w_3) + f_2 - w_0$	$1M$
5	**Compute $\boldsymbol{s' = (k'inv') \bmod u_1}$:** $w_0 = k_0'inv_0', w_1 = k_1'inv_1', s_1' = \tilde{v}_{10}k_1' - \tilde{v}_{11}k_0', s_0' = w_0 - u_{10}w_1$ **If** $s_1' = 0$ **then goto step** 6'	$5M$
6	**Compute $\boldsymbol{s = s_1x + s_0}$ and $\boldsymbol{s_1^{-1}}$:** $w_1 = (rs_1')^{-1}, w_2 = s_1'w_1, w_3 = r^2w_1, s_1 = s_1'w_2, s_0 = s_0'w_2$	$1I, 5M, 1S$
7	**Compute $\boldsymbol{l(x) = su_1 + v_1 = s_1x^3 + l_2x^2 + l_1x + l_0}$:** $l_2 = s_1u_{11} + s_0, l_0 = s_0u_{10} + v_{10}$ $l_1 = (s_1 + s_0)(u_{11} + u_{10}) - s_1u_{11} - s_0u_{10} + v_{11}$	$3M$
8	**Compute $\boldsymbol{u_3 = \mathrm{monic}\left(\frac{f-l^2}{u_1^2}\right) = x^2 + u_{31}x + u_{30}}$:** $u_{30} = w_3(\tilde{v}_{11} + w_3(2u_{11} + s_0^2)), u_{31} = 2s_0 - w_3$	$2M, 1S$
9	**Compute $\boldsymbol{v_3 = -l \bmod u_3 = v_{31}x + v_{30}}$:** $w_1 = u_{31}, u_{31} = w_3u_{31}, w_2 = l_2 - w_1, w_3 = u_{30}w_2$ $v_{31} = (u_{31} + u_{30})(w_2 + s_1) - w_1 - w_3 - l_1, v_{30} = w_3 - l_0$	$3M$
Sum		$1I, 22M, 4S$
6'	**Compute $\boldsymbol{l(x) = su_1 + v_1}$:** $inv = 1/r, s_0 = s_0'inv, l_1 = s_0u_{11} + v_{11}, l_0 = s_0u_{10} + v_{10}$	$1I, 3M$
7'	**Compute $\boldsymbol{u_3 = \mathrm{monic}\left(\frac{f-l^2}{u_1^2}\right) = x + u_{30}}$:** $u_{30} = -(2u_{11} + s_0^2)$;	$1S$
8'	**Compute $\boldsymbol{v_3 = -l \bmod u_3 = v_{30}}$:** $v_{30} = u_{30}(l_1 - u_{30}s_0) - l_0$;	$2M$
Sum		$1I, 14M, 3S$

Table 11. Mixed-Addition Formula on a Genus 2 curve over $\mathbb{F}_p$ (Projective Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_3] = \overline{E}_1 \oplus \overline{E}_2$ $l(x)$ such that $E_1 + E_2 = E_3 + \mathrm{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant $\boldsymbol{r = \mathrm{Res}(u_1, u_2)}$:** $\tilde{U}_{11} = U_{11}Z_2, \tilde{U}_{10} = U_{10}Z_2, z_1 = \tilde{U}_{11} - U_{21}, z_2 = U_{20} - \tilde{U}_{10}$ $z_3 = U_{11}z_1, z_4 = z_2 + z_3, r = z_2z_4 + z_1^2U_{10}$	$5M, 1S$
2	**Compute almost inverse of $\boldsymbol{u_2}$ mod $\boldsymbol{u_1}$:** $inv_1 = z_1, inv_0 = z_4$	–
3	**Compute $\boldsymbol{s'}$:** $w_0 = V_{10}Z_2 - V_{20}, w_1 = V_{11}Z_2 - V_{21}, w_2 = inv_0w_0$ $w_3 = inv_1w_1, s'_1 = z_1w_0 + z_2w_1, s'_0 = w_2 - U_{10}w_3$	$7M$
4	**Precomputations:** $R = r^2, \tilde{s}'_0 = s'_0Z_2, \tilde{s}'_1 = s'_1Z_2, S = s'_1\tilde{s}'_1, \tilde{r} = r\tilde{s}'_1$	$4M, 1S$
5	**Compute $\boldsymbol{l}$:** $l_2 = s'_1U_{21} + \tilde{s}'_0, l_0 = s'_0U_{20} + rV_{20}$ $l_1 = (s'_1 + s'_0)(U_{21} + U_{20}) - s'_1U_{21} - s'_0U_{20} + rV_{21}$	$5M$
6	**Compute $\boldsymbol{U_3}$:** $w_1 = \tilde{U}_{11} + U_{21}, U_{31} = s'_1(2\tilde{s}'_0 - s'_1z_1) - RZ_2, l'_1 = l_1s'_1$ $U_{30} = \tilde{s}'_0(s'_0 - 2s'_1U_{11}) + s_1'^2(z_3 - \tilde{U}_{10} - U_{20}) + 2l'_1 + Rw_1$	$8M, 1S$
7	**Compute $\boldsymbol{V_3}$:** $w_1 = l_2s'_1 - U_{31}, V_{30} = U_{30}w_1 - S(l_0s'_1)$ $V_{31} = U_{31}w_1 + S(U_{30} - l'_1)$	$6M$
8	**Adjust:** $Z_3 = \tilde{r}S, U_{31} = \tilde{r}U_{31}, U_{30} = \tilde{r}U_{30}$	$3M$
Sum		$38M, 3S$

Table 12. Doubling Formula on a Genus 2 Curve over $\mathbb{F}_p$ (Projective Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_3] = [2]\overline{E}_1$ $l(x)$ such that $2E_1 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant and precomputations:** $Z_2 = Z_1^2, \tilde{V}_{11} = 2V_{11}, \tilde{V}_{10} = 2V_{10}, w_0 = V_{11}^2, w_1 = U_{11}^2$ $w_2 = 4w_0, w_3 = \tilde{V}_{10}Z_1 - U_{11}\tilde{V}_{11}, r = Z_2(U_{10}w_2 + \tilde{V}_{10}w_3)$	$5M, 3S$
2	**Compute almost inverse:** $inv'_1 = -\tilde{V}_{11}, inv'_0 = w_3$	–
3	**Compute $\boldsymbol{k'}$:** $w_3 = f_3Z_2 + w_1, w_4 = 2U_{10}$ $\tilde{w}_4 = w_4Z_1, k'_1 = Z_1(2w_1 + w_3 - \tilde{w}_4)$ $k'_0 = U_{11}(2\tilde{w}_4 - w_3) + Z_1(f_2Z_2 - w_0)$	$6M$
4	**Compute $\boldsymbol{s'}$:** $w_0 = k'_0inv'_0, w_1 = k'_1inv'_1$ $s'_1 = Z_1(\tilde{V}_{10}k'_1 - \tilde{V}_{11}k'_0), s'_0 = w_0 - U_{10}w_1$	$6M$
5	**Precomputations:** $R = r^2, \tilde{s}'_0 = s'_0Z_1, \tilde{s}'_1 = s'_1Z_1, S = s'_1\tilde{s}'_1, \tilde{r} = r\tilde{s}'_1$	$4M, 1S$
6	**Compute $\boldsymbol{l}$:** $l_2 = s'_1U_{11} + \tilde{s}'_0, l_0 = s'_0U_{10} + rV_{10}, r' = rV_{11}$ $l_1 = (s'_1 + s'_0)(U_{11} + U_{10}) - s'_1U_{11} - s'_0U_{10} + r'$	$5M$
7	**Compute $\boldsymbol{U_3}$:** $U_{30} = 2(r's'_1 + RU_{11}) + s'_0\tilde{s}'_0, U_{31} = 2s'_0s'_1 - R$	$4M$
8	**Compute $\boldsymbol{V_3}$:** $U_{31} = U_{31}Z_1, w_1 = l_2s'_1 - U_{31}, w_2 = U_{30}w_1$ $V_{31} = U_{31}w_1 + S(U_{30} - l_1s'_1), V_{30} = w_2 - S(l_0s'_1)$	$8M$
9	**Adjust:** $Z_3 = \tilde{r}S, U_{31} = \tilde{r}U_{31}, U_{30} = \tilde{r}U_{30}$	$3M$
Sum		$41M, 4S$