uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Balasingham Balamohan

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Accelerating the Scalar Multiplication on Genus 2 Hyperelliptic Curve Cryptosystems

TITRE DE LA THÈSE / TITLE OF THESIS

Ali Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

E. Kranakis

C. Adams

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

ACCELERATING THE SCALAR MULTIPLICATION ON GENUS 2 HYPERELLIPTIC CURVE CRYPTOSYSTEMS

by
Balasingham Balamohan

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of
the requirements for the degree of

MASTER OF COMPUTER SCIENCE

School of Information Technology and Engineering

at

UNIVERSITY OF OTTAWA

Ottawa, Ontario
December, 2009

Your file Votre référence
ISBN: 978-0-494-61286-6
Our file Notre référence
ISBN: 978-0-494-61286-6

Table of Contents

List of Tables

List of Algorithms

Abstract

Elliptic Curve Cryptography (ECC) was independently introduced by Koblitz and Miller in the eighties. ECC requires shorter sizes of underlying finite fields in comparison to other public key cryptosystems such as RSA, introduced by Rivest, Shamir and Adleman. Hyperelliptic curves, a generalization of elliptic curves, require decreasing field size as genus increases. Hyperelliptic curves of genus g achieve equivalent security of ECC with field size $1/g$ times the size of field of ECC for $g \leq 4$. Recently, a lot of research is being focused on increasing the efficiency of hyperelliptic curve cryptosystems (HECC). The most time consuming operation in HECC is the scalar multiplication. At present, scalar multiplication on HECC over prime fields under performs in terms of computational time compared to ECC of equivalent security. This thesis focuses on optimizing HECC scalar multiplication at the point arithmetic level. At the point arithmetic level we obtain more efficient doubling and mixed addition operations to decrease the computational time in the scalar multiplication using binary expansions of scalars. In addition, we introduce tripling operations for the Jacobians of hyperelliptic curves to make use of multibase representations of scalars that are being used effectively in ECC. We also develop double-add operations for semi-affine coordinates and Lange's new coordinates. We use these double-add operations to improve the computational cost of precomputation for semi-affine coordinates and that of more important main phase of scalar multiplication for semi-affine coordinates and Lange's new coordinates. We derive special addition to improve the cost of precomputation for Lange's new coordinates and projective coordinates.

Acknowledgements

I would like to thank my supervisor Prof. Ali Miri for guidance, encouragement and faith in me. Special gratitude is to Computational Algebra Group of Department of Mathematics and Statistics, University of Sydney for making available an online version of Magma Computational Algebra System [4]. Testing the formulae derived in this paper has increased the confidence in the correctness of the formulae obtained.

Chapter 1

Introduction

In this chapter first we provide the motivation for the problem studied by the thesis and then present a summary of the contributions made and a brief description of contents of the thesis.

1.1 Motivation

With ever increasing use of myriads of versatile communication capable digital devices there is a need for efficient but secure communication. Often two or more parties have to communicate without recourse to a secure channel for exchanging keys. This was not possible until late 1970's when Diffie and Hellman proposed a key exchange protocol [12]. Before this secure communication over electronic channels was possible only when parties to the communication agreed on a key beforehand. In the scheme proposed by Diffie and Hellman, Alice and Bob, wanting to share a key over an insecure channel, agree on group $\mathcal{G}$ of prime order p and a generator g for the group. Alice chooses $x \in \mathbb{Z}_p$ and computes g^x and sends it to Bob over the insecure channel. Bob chooses $y \in \mathbb{Z}_p$ and computes g^y and sends it to Alice over the insecure channel. Now both Alice and Bob can compute $g^{xy} = (g^x)^y = (g^y)^x$. They share a secure key if it is computationally difficult to compute g^{xy} given g^x, g^y and g in the group $\mathcal{G}$. The problem of finding g^{xy} with non negligible probability given g^x and g^y is known as computational Diffie-Hellman problem. A closely related problem which is known as discrete logarithm problem and at least as difficult as the computational Diffie-Hellman problem is the computation of x given g and g^x in a group $\mathcal{G}$. Since the introduction of public key cryptography many schemes were proposed to realize it in practice. One of the earliest systems to have been introduced is RSA scheme due to Rivest, Shamir and Adleman [33]. It is being used successfully for secure communication, ever since the introduction of it. Koblitz [20]

and Miller [26] independently proposed use of elliptic curves for public cryptography in 1985. Hyperelliptic curves are generalizations of elliptic curves. Cryptosystems based on difficulty of Diffie-Hellman problem and discrete logarithm problem can be built based on suitably chosen hyperelliptic curves. In those cryptosystems most time consuming operation is scalar multiplication or exponentiation if the multiplicative notation is used. The study of cryptosystems based on hyperelliptic curves is an active area of research. In this thesis, we propose various techniques at point arithmetic level to speed up scalar multiplication on hyperelliptic curves of genus two over prime fields.

1.2 Hyperelliptic Curve Scalar Multiplication

Most time consuming operation for HECC is the computation kD for a divisor class D on a suitable subgroup of the Jacobian of a hyperelliptic curve for $k \in \mathbb{Z}_n$ where n is the order of the subgroup. Broadly, scalar multiplication involves three levels of abstractions. The top level is representations of scalars. The second level is the point arithmetic level about which the thesis is primarily concerned. For example we may represent the scalar k as a binary number and perform doubling and addition operations to obtain kD. Hence efficient doubling and addition operations at point arithmetic level are required to perform scalar multiplication. The third level can be viewed as a composition of two sublevels. One of them is computation on a polynomial ring over a finite field which is realized by the second sublevel of finite field arithmetic. Each point operation is composed of finite number of field operations.

1.3 Contributions

The contributions of the thesis are as follows:

1. We introduce a new coordinate system which we refer to as semi-affine coordinates.

2. We obtain more efficient traditional point formulae in semi-affine coordinates, Lange's new coordinates and projective coordinates.

3. Improved formulae for double-add operations are obtained for semi-affine coordinates and a new double-add formula for Lange's new coordinates is introduced.

4. We present tripling formulae for semi-affine coordinates, Lange's new coordinates and projective coordinates.

1.4 Overview of Results

In this section, we give an indication of how much improvements one can expect using our methods in various environments. As the relative costs of operation will differ from environment to environment we do a theoretical analysis as well. Reader is referred to Sections 5.5 and 6.5 for details.

1. We improve the precomputation cost by up to 11.4% for semi-affine coordinates compared to that of affine coordinates. Speed up of precomputation for Lange's new coordinates and projective coordinates is up to 12.6%.

2. We obtain up to 11.78% and 13.85% speed up on the scalar multiplication for semi-affine coordinates (in comparison to affine) and Lange's new coordinates respectively

3. Scalar multiplication for projective coordinates is at most 3.3% slower than that of Lange's new coordinates using our formulae.

4. Multibase methods provide speed-up up to 2.35% and 2.9% for semi-affine coordinates and Lange's new coordinates respectively.

1.5 Organization of the Thesis

The organization of this thesis as follows:

1. **Chapter 2:** This chapter introduces the basic concepts of hyperelliptic curves relevant to the scalar multiplication.

2. **Chapter 3:** We discuss the existing number representations that are relevant to this work. In this section, we also present a brief description of various coordinate systems and their significance.

3. **Chapter 4:** In this chapter we describe the methodologies we use to obtain the formulae presented in this thesis.

4. **Chapter 5:** We make a detailed description of formulae for the semi-affine coordinates. We compare the costs using new techniques with those of existing methods.

5. **Chapter 6:** This chapter catalogues formulae for the inversion-free coordinates. We make a comprehensive comparison of improvements made on the scalar multiplication.

6. **Chapter 7:** We conclude with remarks on the thesis and future work.

7. **Appendix A:** For ease of reference, Appendix A is a collection of all the formulae derived in this thesis. We refer to the collection in Chapters 5 and 6.

Chapter 2

Background on Hyperelliptic Curves

In this chapter, we present a brief mathematical background needed to understand computational aspects of hyperelliptic curves relevant to their use in cryptography. This is intended as a brief introduction. For extensive presentations refer to [6, 9, 15, 17, 21, 22, 25].

2.1 Groups

A group $(\mathcal{G}, *)$ is a nonempty set $\mathcal{G}$ with a binary operation satisfying

$$* : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$$

following properties.

1. Associativity: For all $a, b, c \in \mathcal{G}$, $a * (b * c) = (a * b) * c$
2. Identity: There is a i such that for all $a \in \mathcal{G}$, $i * a = a * i = a$
3. Inverse: For all $a \in \mathcal{G}$ there is a $a^{-1} \in \mathcal{G}$ such that $a * a^{-1} = a^{-1} * a = i$

A Group is Abelian, if in addition, for all $a, b \in \mathcal{G}$ $a * b = b * a$. If the binary operation is denoted by $+$, the group is called additive and the terminology familiar from arithmetic is used. Similarly for the case where the operation is denoted $\times$, the group is called multiplicative. A subgroup $(\mathcal{H}, +)$ of a group $(\mathcal{G}, +)$ is:

1. $\mathcal{H} \subset \mathcal{G}$.
2. $(\mathcal{H}, +)$ is a group with the operation $+$ is the operation $+$ on $(\mathcal{G}, +)$ restricted to set $\mathcal{H}$

Let $(\mathcal{G}, +)$ a Abelian group and $(\mathcal{H}, +)$ is a subgroup of $(\mathcal{G}, +)$. Then the quotient group $(\mathcal{Q}, +_Q)$ induced by $(\mathcal{H}, +)$ on $(\mathcal{G}, +)$ is:

1. $\mathcal{Q}$ set of partitions induced on $\mathcal{G}$ by the equivalence $g_1 \sim g_2$ iff $g_1 - g_2 \in \mathcal{H}$ for all $g_1, g_2 \in \mathcal{G}$. We denote by $[g]$ the partition containing g.

2. For $[g_1], [g_2] \in \mathcal{Q}$ the group operation is defined as:

$$[g_1] +_Q [g_2] = [g_1 + g_2].$$

   where $[g]$ represents the partition in which g is an element.

3. The operation is well defined with identity $\mathcal{H}$ as if:

$$[g_1] = [g_2], [g_3] = [g_4]$$

   then

$$[g_1 + g_3] - [g_2 + g_4] = [g_1 + g_3 - g_2 - g_4] = [g_1 - g_2 + g_3 - g_4]$$

   But $g_1 - g_2$ and $g_3 - g_4$ are in $\mathcal{H}$. Hence $[g_1 + g_3] = [g_2 + g_4]$ and identity is $\mathcal{H}$.

2.2 Finite Fields

A field $(\mathbb{F}, +, \times)$ is a nonempty set $\mathbb{F}$ with two binary operations satisfying:

1. Associativity of Addition : For all $a, b, c \in \mathbb{F}$, $a + (b + c) = (a + b) + c$.

2. Additive Identity: There is a $0 \in \mathbb{F}$ such that for all $a \in \mathbb{F}$, $0 + a = a + 0 = a$.

3. Additive Inverse: For all $a \in \mathbb{F}$ there is a $-a \in \mathbb{F}$ such that $a + (-a) = 0$.

4. Commutativity of Addition: For all $a, b \in \mathbb{F}$, $a + b = b + a$.

5. Associativity of Multiplication : For all $a, b, c \in \mathbb{F}$, $a \times (b \times c) = (a \times b) \times c$.

6. Multiplicative Identity: There is a $1 \in \mathbb{F}$ such that for all $a \in \mathbb{F}$, $1 \times a = a$.

7. Multiplicative Inverse: For all $a \in \mathbb{F} - \{0\}$ there is an a^{-1} such that $a \times a^{-1} = 1$.

8. Commutativity of Multiplication: For all $a, b \in \mathbb{F}$, $a \times b = b \times a$.

9. Distributivity Multiplication over Addition : For all $a, b, c \in \mathbb{F}$, $a \times (b + c) = (a \times b) + (a \times c)$.

A finite field is a field with finitely many elements. Every finite field $\mathbb{F}_q$ has $q = p^d$ elements where p is a prime called characteristic of the finite field $\mathbb{F}_q$. If $d = 1$ then the field is classified as a prime field. An extension of a finite field $\mathbb{F}_q$ is a field $\mathbb{F}_{q'}$ such that $\mathbb{F}_q \subset \mathbb{F}_{q'}$. An extension of $\mathbb{F}_q$ always has order $q' = q^r$ for some positive integer r. For a polynomial $f(x)$ of degree n with coefficients in $\mathbb{F}_q$, splitting field of $f(x)$ is the smallest extension of $\mathbb{F}_q$ in which $f(x)$ has n roots. Polynomial $f(x)$ is said to split in any field in which $f(x)$ has n roots. The algebraic closure of a field $\mathbb{F}_q$ is the unique minimal field in which all the polynomial equations, on one variable with coefficients in $\mathbb{F}_q$, split. We denote by $\bar{\mathbb{F}}_q$ the unique algebraic closure of finite field $\mathbb{F}_q$.

2.3 Hyperelliptic Curves

A hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$ is a special point at infinity denoted ∞ and the set of ordered pairs $(x, y) \in \bar{\mathbb{F}}_q \times \bar{\mathbb{F}}_q$ satisfying an equation:

$$C : y^2 + h(x)y = f(x) = x^{2g+1} + \sum_{i=0}^{2g} f_i x^i \tag{2.1}$$

where

1. $h(x) \in \mathbb{F}_q[x]$ is of degree at most g.
2. $f(x) \in \mathbb{F}_q[x]$ is a monic polynomial of degree $2g + 1$.
3. There are no $(x, y) \in \bar{\mathbb{F}}_q \times \bar{\mathbb{F}}_q$ satisfying C and both its partial derivatives.

If the characteristic of the field is not two then $y - \frac{h(x)}{2}$ is well-defined. Application of the replacement of y by $y - \frac{h(x)}{2}$ in Equation 2.1 results in:

$$(y - \frac{h(x)}{2})^2 + h(x)(y - \frac{h(x)}{2}) = f(x)$$

$$y^2 - h(x)y + (\frac{h(x)}{2})^2 + h(x)y - \frac{h(x)^2}{2} = f(x)$$

$$y^2 = f(x) + \frac{h(x)^2}{4}$$

But $deg(h(x)^2) = 2deg(h(x)) \leq 2g$. So the degree of $f(x)+\frac{h(x)^2}{4}$ is a monic polynomial of degree $2g+1$ since $f(x)$ is a monic polynomial of degree $2g+1$. Hence for the case of fields with characteristic not equal to 2 Equation 2.1 reduces to:

$$C : y^2 = f(x) \tag{2.2}$$

If the characteristic, p, of the field does not divide $2g+1$ then $x - \frac{f_{2g}}{2g+1}$ is well-defined. Substituting $x = x - \frac{f_{2g}}{2g+1}$ in the Equation 2.2, we obtain:

$$y^2 = f(x - \frac{f_{2g}}{2g+1})$$

But the coefficient of x^{2g} on right hand side is sum of coefficients of x^{2g} in $(x-\frac{f_{2g}}{2g+1})^{2g+1}$ and f_{2g}. This sum is zero. We observe later in section 2.10 that having $f_{2g} = 0$, speeds-up formulae for $g = 2$.

2.3.1 Technical Point on Point at Infinity

On the projective space there are singular points $(0 : a : 0)$ on a hyperelliptic curve. A technique called normalization is applied to remove singularities. The resulting singular curve agrees with C at the affine points and has a single point at infinity [34]. For our purposes preceding definition of points on hyperelliptic curves is sufficient.

2.4 Divisors on Hyperelliptic Curves

A divisor on a hyperelliptic curve is a finite formal sum (free Abelian group) of points on the curve. That is a divisor D is:

$$D = \sum_{P \in C} m_P P$$

where only finitely many m_P are nonzero. By a formal sum or free Abelian group of points on the curve we mean:

1. Set of elements of the group is:

$$\{\sum_{P\in C, m_P\in\mathbb{Z}} m_P P\}$$

   Where only finitely many m_P are nonzero.

2. Addition is defined as :

$$\sum_{P\in C} m_P P + \sum_{P\in C} n_P P = \sum_{P\in C}(m_P + n_P)P$$

3. Additive identity is the element with $m_P = 0$ for any point $P \in C$.

4. Additive inverse of

$$\sum_{P\in C} m_P P$$

   is

$$\sum_{P\in C}(-m_P)P$$

5. Scalar multiplication for any $k \in \mathbb{Z}$ defined as:

$$k\sum_{P\in C} m_P P = \sum_{P\in C}(km_P)P$$

6. A basis is the set of points on C.

Degree of a divisor

$$\sum_{P\in C} m_P P$$

is defined as

$$\sum_{P\in C} m_P$$

The order of a point in the divisor is the coefficient corresponding to it. A zero divisor is defined as a divisor with zero degree. We denote by Div^0 the subgroup of all degree zero divisors of the vector space of divisors.

2.5 Rational Function

A polynomial function on a hyperelliptic curve, given by equation 2.2, is defined as a polynomial class on two indeterminates x, y under the equivalence relation induced by the equation 2.2. Any polynomial function on a hyperelliptic curve has a unique canonical representation of the form $a(x) - yb(x)$ where $a(x)$ and $b(x)$ are univariate polynomials. This follows from the fact that we can replace y^k for $k \geq 2$ by $y^{k-2}f(x)$. Applying this substitution repeatedly we can obtain any polynomial on x and y in the above form. Uniqueness follows from the observation two representations of the same polynomial function should differ by a $\alpha(x, y)(y^2 - f(x))$ for some polynomial $\alpha(x, y)$. Since the difference of two representations have degree with respect to y at most 1 it should be that $\alpha(x, y) = 0$. Degree of a polynomial function in its canonical form $P = a(x) - yb(x)$ is the maximum of two times the degree of $a(x)$ as a polynomial in x and $2g + 1 + 2d$ where d is the degree of $b(x)$ as a polynomial in x.

A rational function on a hyperelliptic curve is defined as

$$R(x, y) = \frac{P(x, y)}{Q(x, y)}$$

where $P(x, y)$ and $Q(x, y)$ are polynomial functions on the hyperelliptic curves such that $R(x, y)$ is defined for at least one point of the hyperelliptic curve C other than the point at infinity. The value of a rational function $R(x, y) = P(x, y)/Q(x, y)$ at the point at infinity is as follows:

1. Zero if degree of the $P(x, y)$ is less than that of the $Q(x, y)$.

2. The ratio of the coefficients of the highest degrees if the degrees of both $P(x, y)$ and $Q(x, y)$ are equal.

3. Undefined in other cases.

Divisor of a rational function on a hyperelliptic curve is defined as

$$Div(R(x, y)) = \sum_{i=1}^{k_z} m_i P_i - \sum_{i=1}^{k_p} n_i Q_i$$

where $P_i's$ are zeroes of $R(x,y)$ with multiplicity m_i and $Q_i's$ are poles of $R(x,y)$ with multiplicity n_i. The degree of divisor corresponding to $R(x,y)$, $\sum_{i=1}^{k_z} m_i - \sum_{i=1}^{k_p} n_i$ is always zero by the definition of the value of a rational function at the point at infinity. The set of rational functions on a hyperelliptic curve has following properties.

1. The product of any two rational functions not necessarily distinct is also a rational function. Hence, the set of divisors corresponding to a rational function is closed under group operations.

2. The divisor corresponding to a constant function is the identity element of the free Abelian group.

Hence, the set of divisors corresponding to rational functions is a subgroup of the free Abelian group of divisors and particularly of Div^0. This subgroup is known as the set of Principal divisors denoted by Div^p.

The quotient group $\mathcal{J}_C = Div^0/Div^p$ is called the Jacobian of the hyperelliptic curve which is also known as divisor class group. We denote the equivalence induced by the quotient group by $\sim$.

2.6 Semi-reduced and Reduced Divisors

A semi-reduced divisor is a divisor of the form:

$$\sum_{i=1}^{r} m_i P_i - (\sum_{i=1}^{r} m_i)\infty \tag{2.3}$$

Such that all $m_i \geq 0$ and if a point and its involution are distinct at most one of them has nonzero order and if they are not distinct and the point is not ∞ the point has as order 1. Every divisor D has a semi-reduced divisor D' such that $D \sim D'$. If the semi-reduced divisor satisfies the property that $\sum_{i=1}^{r} m_i \leq g$ then the divisor is called a reduced divisor. For any divisor D there exists a unique reduced divisor D'' such that $D \sim D''$, i.e an element of the $\mathcal{J}_C$ can be uniquely represented by a reduced divisor.

2.7 Polynomial Representation of a Reduced Divisor

Mumford [31] showed that a reduced divisor can be represented uniquely by two polynomials $U(X), V(X) \in \bar{\mathbb{F}}_q[X]$ where:

1. $U(X)$ is monic,
2. $deg(V(X)) < deg(U(X)) \leq g$,
3. $U(X)|(f(X) - V(X)h(X) - V(X)^2)$.
4. $U(x_i) = 0$ for all points $P_i = (x_i, y_i)$ with positive order in the reduced divisor.
5. $V(x_i) = y_i$ for all points $P_i = (x_i, y_i)$ with positive order in the reduced divisor.

This follows that if (U, V) represent a reduced divisor $P_1 + P_2 - 2\infty$ then $(U, -V)$ represents its additive inverse in the divisor class group. Following arguments show that for the case degree of U is 2. Arguments for the cases when degree of U is 1 or 0 are similar.

1. U is a rational function with zeroes at $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = (x_1, -y_1)$ and $P_4 = (x_2, -y_2)$.
2. U also has as pole ∞ with order 4.
3. $-V(x_i) = -y_i$ for $i = 1, 2$.

Cantor presented an algorithm to perform group addition in the Mumford representation [6]. Cantor's algorithm is presented as Algorithm 1. Hyperelliptic curves, having algebraic structures, can be used in protocols based on the discrete logarithm problem. Their geometric structure leads to polynomial representations, hence the scalar multiplication can be efficiently implemented on hardware and software.

2.8 Genus 2 Hyperelliptic Curves over Prime Fields in Cryptography

Divisor classes represented by $U(X), V(X) \in \mathbb{F}_q[X]$ form a finite Abelian subgroup $\mathcal{J}_C(\mathbb{F}_q)$ of the divisor class group. To build secure systems using $\mathcal{J}_C(\mathbb{F}_q)$ of hyperelliptic curves, $\mathcal{J}_C(\mathbb{F}_q)$ should have a large enough prime factor in it's order. Order N_q

Algorithm 1 Cantor's Algorithm for Group Law (arbitrary genus g, $\mathbb{F}_p$)

Input: $D_1 = (u_1, v_1), D_2 = (u_2, v_2), C : y^2 = f(x)$
Output: $D_3 = (u_3, v_3)$ reduced with $D_3 = D_1 + D_2$
1. $d_1 = \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2. $d = \gcd(d_1, v_1 + v_2) = c_1 d_1 + c_2(v_1 + v_2)$
3. $s_1 = c_1 e_1$, $s_2 = c_1 e_2$, $s_3 = c_2$
4. $u = \frac{u_1 u_2}{d}$, $\frac{s_1 u_1 v_1 + s_2 u_2 v_1 + s_3(v_1 v_2 + f(x)}{d}$
5. **repeat**
6. $\quad u' = \frac{f(x) - v^2}{u}$, $v' = -v \pmod{u'}$
7. $\quad u = u'$, $v = v'$
8. **until** $deg(u) \leq g$
9. $u_3 =$ monic form of u', $v_3 = v'$

of the $\mathcal{J}_C(\mathbb{F}_q)$ a of hyperelliptic curve over $\mathbb{F}_q$ satisfies following inequality:

$$(q^{1/2} - 1)^{2g} \leq N_q \leq (q^{1/2} + 1)^{2g} \tag{2.4}$$

If k-bit field results in a group of order n_k for elliptic curves and $\frac{k}{2}$-bit field results in a group of order $n_{\frac{k}{2}}$ for hyperelliptic curves of genus 2 then:

$$n_k \approx (2^{\frac{k}{2}})^2 = 2^k$$

and

$$n_{\frac{k}{2}} \approx (2^{\frac{k}{4}})^4 = 2^k$$

This implies that it is sufficient to use appropriately chosen hyperelliptic curve of genus two over field of size half of that of elliptic curves for equivalent security. Cantor's algorithm is not efficient enough to be practical. Harley published very efficient algorithms distinguishing cases and eliminating redundant computations [15, 17]. More details on Cantor's and Harley's algorithms can be found in [9, 22, 35]. Harley's algorithms for doubling operation and addition are presented in Algorithms 2 and 3. Explicit formulae derived from Harley's algorithms [22] for the frequent cases are presented in Tables 2.1 and 2.2.

2.9 Different Cases

Almost always, the polynomial U representing a divisor is of degree two whereas polynomial V is of degree one with $gcd(U, V) = 1$ for curves with cryptographically useful parameters. For the frequent case of addition of (U_1, V_1) and (U_2, V_2), $\gcd(U_1, U_2) = 1$. If $s_1 = 0$ then U polynomial of the result is of degree one where $S = s_1 x + s_0$ (Algorithm 2). Usually s_1 is nonzero. We consider only the frequent cases in our discussions. Less frequent cases are analyzed extensively in [14] and [22]. Frequent cases almost always occur for curves with field size useful for security applications. We observe that tests for categorization of the cases are performed as a part of the formulae hence categorization does not require additional operations. A cryptosystem may be implemented without dealing with infrequent cases by implementing it to choose another random scalar and perform the protocol whenever infrequent cases occur without performance penalty on average. We also assume $h(x) = 0$ and the coefficient of x^4 in $f(x)$ is zero.

2.10 Explicit Formulae from Harley's Algorithms

Explicit formulae derived from Harley's algorithms [22] for the frequent cases are presented in Tables 2.1 and 2.2.

2.10.1 Doubling Formula

We give a description of doubling formula presented in Table 2.1. Reader is referred to Harley's algorithm for doubling (Algorithm 3 in page 16) as well.

1. Step 1: $2V_1$ is computed.
2. Step 2: Resultant r of V_1 and U_1 is computed. $r = 0$ then $\gcd(U_1, U_2)$ is not one and the case is not the frequent one. The resultant r is the determinant of the linear system obtained by the following polynomial identity:

$$2V_1(i_1 X + i_0) - i_1(2v_{11})U_2 = 1$$

3. Step 3: Pseudo-inverse $I = i_1 X + i_0 = (r/2V_1) \pmod{U_1}$ is computed.

4. Step 4: Exact division of $(f(X) - V_1^2)$ by U_1 followed by modular reduction by U_1. We observe that having $f_4 = 0$ speed up formula.

5. Step 5: $S' = rS$ is computed. If $s_1' = 0$ then resulting divisor will have U polynomial with degree strictly less than two.

6. Step 6: Monic form S'' of $S = s_1X + s_0 = K/2V_1 \pmod{U_1}$ for obtaining U_3. and s_1 required for deriving V_3 is obtained.

7. Step 7: Monic form L' of L in Harley's algorithm for doubling Algorithm is computed (Algorithm 3 of page 16).

8. Step 8: U_3 is computed by observing that we are required to compute only the first three leading coefficients of monic form of $K - S(SU_1 + 2V_1)$. This step for doubling is more efficient than that of addition by the fact that $U_3 = S''^2 + \frac{2V_1 - K}{s_1^2 U_1}$.

9. Step 9: V_3 is computed as $-V_1 + s_1(-L' \pmod{U_3})$ for efficiency.

2.10.2 Addition Formula

We give a description of addition formula presented in Table 2.2. Reader is referred to Harley's algorithm for addition (Algorithm 2 in page 16) as well.

1. Step 1: Resultant r of U_1 and U_2 is computed. If $r = 0$ then $\gcd(U_1, U_2)$ is not one and the case is not the frequent one. The resultant r is the determinant of the linear system obtained by the following polynomial identity:

$$(U_1 - U_2)(i_1X + i_0) - i_1(u_{11} - u_{21})U_2 = 1$$

2. Step 2: Pseudo-inverse $I = i_1X + i_0 = (r/(U_1) \pmod{U_2}$ is computed.

3. Step 3: $S' = rS = r\frac{V_2 - V_1}{U_1} \pmod{U_1}$ is computed. If $s_1' = 0$ then resulting divisor will have U polynomial with degree strictly less than two.

4. Step 4: Since monic S'' form of $S = s_1X + s_0 = K/2V_1 \pmod{U_1}$ for obtaining U_3 and s_1 required for deriving V_3 is obtained.

5. Step 5: Monic form L' of L ,in Harley's algorithm for addition Algorithm, is computed (Algorithm 2 of page 16).

6. Step 6: U_3 is computed by observing that we are required to compute only the first three leading coefficients of monic form of $K - S(SU_1 + 2V_1)$.

7. Step 7: V_3 is computed as $-V_1 + s_1(-L' \pmod{U_3})$ for efficiency.

Algorithm 2 Harley's Algorithm for Group Addition ($g = 2$, $\mathbb{F}_p$, $p > 2$ a prime, $h = 0$)

Input: $D_1 = (U_1, V_1), D_2 = (U_2, V_2), C : Y^2 = f(X)$
Output: $D_3 = (U_3, V_3)$ reduced with $D_3 = D_1 + D_2$
1. $K = \frac{f(X)-V_1^2}{U_1}$ (division)
2. $S = s_1X + s_0 = \frac{V_2-V_1}{U_1} \pmod{U_2}$
3. $L = SU_1$
4. $U_3 = \frac{K-S(L+2V_1)}{U_2}$ (division)
5. make U_3 monic
6. $V_3 = -(L + V_1) \pmod{U_3}$

Algorithm 3 Harley's Algorithm for Group Doubling ($g = 2$, $\mathbb{F}_p$, $p > 2$ a prime, $h = 0$)

Input: $D_1 = (U_1, V_1) C : Y^2 = f(X)$
Output: $D_2 = (U_2, V_2)$ reduced with $D_2 = 2D_1$
1. $K = \frac{f(X)-V_1^2}{U_1}$ (division)
2. $S = s_1X + s_0 = \frac{K}{2V_1} \pmod{U_1}$
3. $L = SU_1$
4. $U_2 = \frac{K-S(L+2V_1)}{U_1}$ (division)
5. make U_2 monic
6. $V_2 = -(L + V_1) \pmod{U_2}$

2.11 Summary

In this chapter, we presented a brief discussion of the algebraic aspects of hyperelliptic curves relevant to their use in cryptography. Efficiency of scalar multiplication depends on the representation of scalars in addition to the efficiency of group operations. In the next chapter, we briefly review material relevant to representations of scalars.

Table 2.1: Doubling Formula for Affine Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ (U_1, V_1) where $D_1 = (U_1, V_1)$ is a reduced divisor $U_1 = X^2 + u_{11}X + u_{10}$, $V_1 = v_{11}X + v_{10}$	
	Output:$D_3 = 2D_1$ where $D_3 = (U_3, V_3)$ is a reduced divisor $U_3 = X^2 + u_{31}X + u_{30}$,$V_3 = v_{31}X + v_{30}$	
Step	Expression	Cost
1	Compute $V' = 2V_1 \pmod{U_1}$ $v_1' = 2v_{11}, v_0' = 2v_{10}$	-
2	Compute resultant r of V' and U' $w_0 = v_{11}^2, w_1 = u_{11}^2, w_2 = 4w_0, w_3 = u_{11}v_1'$ $r = u_{10}w_2 + v_0'(v_0' - w_3)$ if $r = 0$ then different case	3M+2S
3	Compute pseudo-inverse $I = i_0 + i_1x = r/2V' \pmod{U_1}$ $i_1 = -v_1', i_0 = v_0' - w_3$	-
4	Compute $k = (f - V^2)/U_1 \pmod{U_1} = k_1X + k_0$ $w_3 = f_3 + w_1, w_4 = 2u_{10}, k_1 = 2w_1 + w_3 - w_4$ $k_0 = u_{11}(2w_4 - w_3) + f_2 - w_0$	1M
5	Compute $S' \equiv kI \pmod{U_1}$ $w_0 = k_0i_0, w_1 = k_1i_1$ $s_1' = (i_0 + i_1)(k_0 + k_1) - w_0 - w_1(1 + u_{11})$ $s_0' = w_0 - u_{10}w_1$ if $s_1' = 0$ different case	5M
6	Compute $S'' = X + s_0/s_1 = X + s_0'/s_1'$ and s_1 $w_0 = 1/(s_1'r)$,$w_1 = w_0r$,$w_2 = w_0s_1'^2 = s_1$ $w_3 = rw_1(= 1/s_1)$,$w_4 = w_3^2, s_0'' = s_0'w_1$	I+5M+2S
7	Compute $L'' = S''U_1 = X^3 + l_2'X^2 + l_1'X + l_0'$ $l_2' = u_{11} + s_0'', l_1' = u_{11}s_0'' + u_{10}, l_1' = u_{10}s_0''$	2M
8	Compute $U_3 = (S(L + 2V_1) - K)/U' = X^2 + u_{31}X + u_{30}$ $u_{30} = s_0''^2 + 2v_1w_3 + 2u_1w_4$ $u_{31} = 2s_0'' - w_4$	2M+1S
9	Compute V_3 $w_0 = l_2' - u_{31}, w_1 = u_{31}w_0 + u_{30} - l_1', v_{31} = w_1w_2 - v_{11}$ $w_1 = u_{30}w_0 - l_0', v_{30} = w_1w_2 - v_0$	4M
	Total Cost : I+22M+5S	

Table 2.2: Addition Formula for Affine Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ (U_1, V_1) where $D_1 = (U_1, V_1)$ is a reduced divisor reduced divisor $D_2 = (U_2, V_2)$ $U_1 = X^2 + u_{11}X + u_{10}$, $V_1 = v_{11}x + v_{10}$ $U_2 = X^2 + u_{21}X + u_{20}$, $V_1 = v_{21}x + v_{20}$	Cost
	Output: (U_3, V_3) where $D_3 = (U_3, V_3) = D_1 + D_2$ is a reduced Divisor $U_3 = X^2 + u_{31}X + u_{30}, V_3 = v_{31}X + v_{30}$	
Step	Expression	Cost
1	Compute the resultant r of U_1, U_2 : $t_1 = u_{11} - u_{21}, t_0 = u_{20} - u_{10}, t_2 = t_1u_{11} + t_0$, $r = t_2t_0 + t_1^2u_{10}$ if $r = 0$ then different case	4M+1S
2	Compute almost inverse of U_2 modulo U_1 $i_1 = t_1, i_0 = t_2$	2M
3	Compute $S' = s_1'X + s_0' \equiv r(V_1 - V_2)/U_1 \pmod{U_2}$: $w_0 = v_{10} - v_{20}, w_1 = v_{11} - v_{21}, w_2 = i_0w_0, w_3 = i_1w_1$ $s_0' = w_2 - u_{10}w_3, s_1' = (i_0 + i_1)(w_0 + w_1) - w_2 - w_3(1 + u_{11})$; If $s_1' = 0$, different case	4M
4	Compute $S'' = X + s_0/s_1 = X + s_0'/s_1'$, and $1/s_1$ $w_0 = (rs_1')^{-1}$,$w_1 = rw_0$,$w_2 = s_1^{'2}w_0$ $w_3 = rw_1$,$w_4 = w_3^2$,$s_0^{''} = s_0'w_1$	1I+3M+2S
5	Compute $L'' = S''U_1 = X^3 + l_2'X^2 + l_1'X + l_0'$ $l_2' = u_{21} + s_0^{''}, l_1' = u_{21}s_0^{''} + u_{20}, l_1' = u_{10}s_0^{''}$	2M
6	Compute $U_3 = (S(L + 2V_1) - K)/U_2 = X^2 + u_1'X + u_0'$: $u_{30} = (s_0^{''} - u_{11})(s_0^{''} - t_1) - u_{10} + l_1' + 2v_{21}w_3 + (u_{11} + u_{21})w_4$ $u_{31} = 2s_0^{''} - t_1 - w_4$	3M
7	Compute V_3 $w_0 = l_2' - u_{31}, w_1 = u_{31}w_0 + u_{30} - l_1', v_{31} = w_1w_2 - v_{21}$ $w_1 = u_{30}w_0 - l_0', v_{30} = w_1w_2 - v_{20}$	4M
	Total Cost: I+22M+3S	

Chapter 3

Number Representations for Efficient Scalar Multiplication

In this chapter, we introduce number representations used to decrease computational time of scalar multiplication on the Jacobian of hyperelliptic curves.

3.1 Binary Representation

This form of representation and its usage for scalar multiplication has been known for more than 2000 years [9]. We present left-to-right and right-to-left binary scalar multiplication algorithms as Algorithms 4 and 5. The Number of doubling operations is approximately $log_2(k)$ for both algorithms. The expected average number of additions is the average Hamming weight of the binary representation and is $\frac{1}{2}log_2(k)$.

Algorithm 4 Left-to-Right Binary Scalar Multiplication

Input: An element D of a group $\mathcal{G}$ and a non-negative integer $k = (k_{l-1} \cdots k_0)_2$
Output: The element $kD \in \mathcal{G}$
1: $D' = 1$, $i = l - 1$
2: **while** $i \geq 0$
3: $\quad D' = 2D$
4: $\quad$ **if** $k_i = 1$ **then** $D' = D + D'$
5: $\quad i = i - 1$
6: **end while**
7: **return** D'

3.2 Non Adjacent Form (NAF) and Multibase Representations

An odd integer is either 1 (mod 4) or -1 (mod 4). The additive inverse of divisor class represented in Mumford form as (U, V) is $(U, -V)$. Preceding fact can be utilized to decrease the number of additions in scalar multiplication without precomputation. General idea of this method was introduced by Brauer [5]. The idea is

Algorithm 5 Right-to-Left Binary Scalar Multiplication

Input: An element D of a group $\mathcal{G}$ and a non-negative integer $k = (k_{l-1} \cdots k_0)_2$
Output: The element $kD \in \mathcal{G}$
1: $D' = 1$, $D'' = D$, $i = 0$
2: **while** $i \leq l - 1$
3: **if** $k_i = 1$ **then** $D' = D' + D''$
4: $D'' = 2D''$;$i = i + 1$
5: **end while**
6: **return** D'

to represent a positive integer on a base 2^w where w is called window size. If an integer k is $a \pmod{2^w}$ then it is also $(a - 2^w) \pmod{2^w}$. Hence windowed-NAF representation of window size w can be used for scalar multiplication on Jacobians of a hyperelliptic curves using only 2^{w-2} precomputed values. Bosma [3] made precise analysis of density for $w = 2$. The density is 1/3 on average. In general, windowed NAF with window size w has density $\frac{1}{w+1}$ on average. More details of NAF and windowed-NAF can be found in [1–3,5,7,8,10,18,29,30,32]. Longa and Miri [24] generalized this method for more than one base. We present their most general recoding algorithm and corresponding scalar multiplication algorithm in Algorithms 6 and 7 respectively. They refer to their representation as extended windowed multibase NAF ($ex\text{-}wmbNAF$). Longa and Miri [24] showed that addition is performed only if the scalar is not divisible by any b_j for all j such that $w_j > 0$. Hence the precomputed points needed are $\{nP : 1 \leq n \leq b/2$ and $b_j \nmid n$ for all $1 \leq j \leq J$ such that $w_j > 0\}$ where $b = \prod_{i=1}^{n} b_i^{w_i}$.

w-NAF corresponds to base set of singleton 2 with weight w. We denote by $(b_1, b_2, \cdots b_l)$-$ex\text{-}wmbNAF_{w_1,w_2,\cdots w_l}$ with base set $\{b_1, b_2, \cdots b_l\}$ with each b_i having weight w_i. Longa and Miri use the terminology $wmbNAF$ for the case with $b_1 = 2$ and $w_i = 0$ for $i \neq 1$. We use only restricted case of these representations with $l = 2$, $b_1 = 2$, $b_2 = 3$ and $w_2 = 0$ and we denote that case by $(2,3)$-$wmbNAF_{w_1}$ where w_1 is the weight of base two. Using Markov chains, [27] Li and Miri established the nonzero density and the average number of times each composite operation is performed. Table 3.1 presents the average number of doubling operations, tripling operations and additions needed to be performed for $(2,3) - wmbNAF_w$ for 256-bit

scalar.

Algorithm 6 Algorithm For *ex-wmbNAF* Multibase Recoding

Input: Scalar k
Bases $B = (b_1, b_2, \cdots, b_J)$, is an ordered set of prime numbers
Window set $W = \{w_1, w_2, \cdots, w_J\}$ is an ordered set of non-negative integers
$b = \prod_{j=1}^{j=J} b_j^{w_j}$
Output:$k = (k_l, k_{l-1}, \cdots, k_0)$. $k_i = (d_i, r_i)$ where $|d_i| < \prod_{j=1}^{j=J} b_j^{w_j}$
b_j does not divide d_i for all $1 \leq j \leq J$ such that $w_j > 0$,
$r_i \in B$ and $k = \sum_{j=0}^{j=l} d_j \prod_{m=0}^{m<j} r_m$
1.$i = 0$
2.While $k > 0$ **do**
3. **Find the first** j **such that** $b_j | k$ **and set** $r_i = b_j, d_i = 0, k = k/b_j$
4. **If there is no** j **such that** $k \pmod{b_j} \equiv 0$ $1 \leq j \leq J$
5. $d_i = k \pmod{b}$
6. **If** $(d_i > w/2)$ **then**
7. $d_i = b_i - b$
8. **end if**
9. $k = k - d_i$
10. **end if**
11. $k_i = (d_i, r_i); i = i + 1;$
12.end while
13.Return $(k_{i-1}, k_{i-2}, \cdots, k_1)$

Algorithm 7 Algorithm for Extended *wmbNAF* Method for Scalar Multiplication

Input: P, $(k_l, k_{l-1}, \cdots, k_0)$ and $d_i P$ where $k_i = (d_i, r_i)$ $0 \leq i \leq l$
Output: kP where $k = \sum_{j=0}^{j=l} d_j \prod_{m=0}^{m<j} r_m$
1.$Q = O$
2.For $i = l$ **down to** 0 **do**
3. $Q = r_i Q + d_i P$
4.end for
5.Return Q

3.3 Different Systems of Coordinates

Field inversions are necessary for obtaining results of point operations. But they are hard to implement in hardware and have high computational cost in software implementations. Weighted coordinate systems are used to eliminate inversion of

Table 3.1: Number of Different Operations for 256-Bit Scalar using $(2,3)-wmbNAF_w$

Weight of 2(w)	**No. of Doublings**	**No. of Triplings**	**No. of Additions**
2	202.5	33.7	67.5
3	213.7	26.7	53.4

field elements in the point operations. A reduced divisor (U, V) is represented in various coordinates as follows:

1. Affine Coordinates : (U, V)
2. Semi-Affine Coordinates (Introduced in this work): $(U, ZV, Z, z = Z^2)$
3. Lange's New Coordinates:$(Uz_1, VZ_1^3Z_2, Z_1, Z_2, z_1 = Z_1^2, z_2 = Z_2^2)$
4. Projective Coordinates:(UZ, VZ, Z)

3.4 Summary

In the previous chapter, we reviewed algebraic aspects of hyperelliptic curves. In this chapter, we presented a discussion on representation and manipulation of scalars. Having presented the necessary background, in the subsequent chapter, we describe the existing and new techniques we use to obtain efficient point formulae.

Chapter 4

Description of Techniques Used In Deriving the Formulae

In this chapter, we describe the existing techniques and the new techniques we use to obtain point operations in Chapters 5 and 6. Computation on the Jacobian of a hyperelliptic is performed using Mumford representation of its elements. Mumford representation of an element of the Jacobian is a pair of polynomials. Hence group operations on the Jacobian is efficiently performed using modern methods of computer algebra. Systematic techniques achieving asymptotically tighter bound are used both directly and by adapting to the special case of small degree polynomial computation arising in the practical hyperelliptic curve cryptosystems. First we survey the existing techniques: Karatsuba multiplication, Montgomery simultaneous inversions, techniques for efficient polynomial division and side-stepping reduction of intermediate V polynomial for the computation of double-add and tripling operations. Next we introduce new techniques: Computation of modular fraction, computing V with weight instead of exactly, computing the U polynomial of the result of tripling operation modulo the U polynomial of the divisor class being tripled.

4.1 Existing Techniques

4.1.1 Karatsuba Multiplication

Karatsuba multiplication of polynomials, introduced in [19], is a technique now widely used to trade more expensive field multiplications for less expensive field additions. The product of a pair of degree one polynomials $ax + b$ and $cx + d$ is computed as follows [22]:

$$(ax + b)(cx + d) = abx^2 + ((a + b)(c + d) - ab - cd)x + cd$$

This saves one multiplication at the expense of two subtractions and one addition. Similar technique is applied to reduce a degree three polynomial modulo degree two polynomial. We perform that computation as follows:

$(ax^3 + bx^2 + cx + d)$

$\equiv (c - (u_1 + u_0)(a + (b - u_1 a)) + u_1 a + u_0(b - u_1 a))x + d + u_0(b - u_1 a) \pmod{U}$

where

$U = x^2 + u_1 x + u_0$

4.1.2 Montgomery's Simultaneous Inversions

Montgomery [28] introduced the following method for inverting simultaneously n field elements $\{a_i : 1 \leq i \leq n\}$:

1. Compute $A_r = \prod_{i=1}^{r} a_i$
2. Invert A_n to obtain I_n
3. Repeatedly compute $i_r = a_r^{-1}$ by computing from $r = n$ to $r = 2$:
   (a) $i_r = A_{r-1} I_r$
   (b) $I_{r-1} = a_r I_r$
   (c) $r = r - 1$

We perform $n - 1$ multiplications to compute A_r. The cost for computing I_r is one inversion and further $n - 1$ multiplications and cost for computing i_r is $n - 1$ multiplications. So the total cost is $1I + 3(n - 1)M$ on average where I denotes the average cost of the inversion and M stands for the average cost of the multiplication.

4.1.3 Exact Division by a Polynomial

Exact division of a degree three polynomial by a monic polynomial of degree two is performed as follows [22]:

1. We obtain the polynomial equation:

$$(ax^3 + bx^2 + cx + d) = (q_1 x + q_0)(x^2 + u_1 x + u_0)$$

2. By equating coefficients we obtain:

$$q_1 = a$$

$$q_0 = b - q_1 u_1$$

This method can be extended to division of degree five polynomial as follows.

$$(ax^5 + bx^4 + cx^3 + dx^2 + ex + f) = (q_3 x^3 + q_2 x^2 + q_1 x + q_0)(x^2 + u_1 x + u_0)$$

$$q_3 = a$$

$$q_2 = b - q_3 u_1$$

$$q_1 = c - q_3 u_0 - q_2 u_1 = c - (q_3 + q_2)(u_0 + u_1) + q_3 u_1 + q_2 u_0$$

$$q_0 = d - q_2 u_0 - q_1 u_1$$

4.1.4 Computation of Double-Add and Tripling without Computing Intermediate V

In [13] Eisenträger, Lauter and Montgomery introduced a technique by which double-add operation can be sped up by not computing $y-$coordinate of intermediate result of double-add operation while performing double-add operation as two additions for the elliptic curves. Fan and Gong [14] adapted this technique to genus two hyperelliptic curves over prime fields in the affine case. We use that technique for semi-affine as well as inversion-free coordinates. Following lemma is central to their method.

Lemma 1. *Let C be a genus 2 HEC over $\mathbb{F}_q$ given by,*

$$C : Y^2 = f(X) \tag{4.1}$$

where $f(X) = X^5 + f_4X^4 + f_3X^3 + f_2X^2 + f_1X + f_0 \in \mathbb{F}_q[X]$. Assume that $D_1 = [U_1, V_1], D_2 = [U_2, V_2]$ and $D' = [U', V'] = D_1 + D_2$ are reduced divisor classes in the Jacobian $\mathcal{J}_C(\mathbb{F}_q)$ of C and satisfy that U_1, U_2 and U' are quadratic, and gcd(U_1, U_2)=gcd(U_1, U')=1. Let S and $\tilde{S}$ satisfy the congruent relations: $S \equiv \frac{V_2 - V_1}{U_1}$

$(\text{mod } U_2)$ *and* $\tilde{S} \equiv \frac{V'-V_1}{U_1} \pmod{U'}$, *then the following modular identity holds.*

$$\tilde{S} \equiv -S - \frac{2V_1}{U_1} \pmod{U'}. \tag{4.2}$$

Using Lemma 1, we obtain efficient double-add and tripling operation.

4.2 Techniques Introduced In This Thesis

4.2.1 Computation of Fraction of Polynomials Modulo another Monic Polynomial

To compute $S = s_1x + s_0 = (ax + b)/(cx + d) \pmod{x^2 + u_1x + u_0}$, we proceed as follows

1. We obtain a polynomial equation for S

$$ax + b = (cx + d)(s_1x + s_0) - s_1(cx^2 + cu_1x + cu_0)$$

2. By equating the coefficients, we obtain the following linear system on s_0 and s_1

$$ds_0 - cu_0s_1 = b$$

$$cs_0 + (d - cu_1)s_1 = a$$

   We use two multiplications for computing coefficients.

3. We use the Cramer's rule to to solve the linear system up to a factor of determinant at a cost of six multiplications.

Traditionally, resultant method is used to obtain the pseudo-inverse I of $cx + d$ up to the factor of resultant r at a cost of three multiplications and a squaring operation [22]. rS is obtained by using Karatsuba multiplication of I and $ax + b$ followed by modular reduction at a cost of five multiplications. Our method saves one squaring operation.

4.2.2 Computing V with a weight

In the Mumford representation (U, V) of divisor D, U is always monic and V is almost always not monic. Hence computing U up to a constant substantially increases the multiplications needed. We observe from Harley's algorithms presented in Algorithms 2 and 3 that computation of the first polynomial of Mumford representation does not require the values of s_1 or s_0 where $S = s_1 x + s_0$. Also computation of $(1/s_1)$ times the second polynomial V also does not require the values of s_1 or s_0. Hence, we use a technique of computing $\bar{V} = V/s_1$ instead of V of the result of addition or doubling operation. This introduces extra three multiplications for normalization. But we save on the overhead involved in computing two inversions needed for computing $s_1, s_0, s_0/s_1, 1/s_1$ using Montgomery's simultaneous inversions [28]. Further saving is introduced by the simplified modular reductions of $S'' U_1 + \bar{V}_1$ where S'' is the monic form of S.

4.2.3 Computing U_3 modulo U_1 and adjusting U_3

We observe that in the Harley's algorithm for addition and doubling used to obtain tripling:

$$s_1^2 U_1 U_2 = K - S(SU_1 + 2V_1) \tag{4.3}$$

and

$$\tilde{s}_1^2 U_2 U_3 = K - \tilde{S}(\tilde{S}U_1 + 2V_1) \tag{4.4}$$

where

1. $D_1 = (U_1, V_1)$ is the divisor being tripled
2. $D_2 = (U_2, V_2) = 2D_1$
3. $D_3 = (U_3, V_3) = 3D_1$
4. $K = \frac{f(X) - V_1^2}{U_1}$
5. $S = \frac{K}{2V_1} \pmod{U_1}$
6. $\tilde{S} = \frac{V_2 - V_1}{U_1} \pmod{U_2}$

We subtract 4.3 from 4.4 to obtain:

$$\tilde{s}_1^2 U_2 U_3 - s_1^2 U_1 U_2 = S(SU_1 + 2V_1) - \tilde{S}(\tilde{S}U_1 + 2V_1) \tag{4.5}$$

Hence we obtain in the case of $\gcd(U_1, U_2) = 1$:

$$\tilde{s}_1^2 U_3 = \frac{2(S - \tilde{S})V_1}{U_2} \pmod{U_1} \tag{4.6}$$

In the case of inversion free coordinates, normalization results in several multiplications in the last step of computation of U_3. We observe that computing a weighted $u_{30} - u_{10}$ using the above identity saves on that overhead.

4.3 Summary

This chapter surveyed existing techniques and introduced new techniques at the point arithmetic level used in the thesis to speed up scalar multiplication on the Jacobian of hyperelliptic curves. One of the techniques, computing weighted V, leads to a coordinate system which we refer as semi-affine coordinates. This is the simplest coordinate system apart from affine coordinate system. In the next chapter we derive formulae for semi-affine coordinate system and compare the cost with new methods for semi-affine coordinates against existing methods for affine coordinates.

Chapter 5

Scalar Multiplication on Semi-Affine Coordinates

In this chapter, we present more efficient traditional formulae and new composite formulae in semi-affine coordinates using techniques presented in Chapter 4. The coordinate system we use is slightly different from the traditional affine coordinates. We represent a reduced divisor (U, V) in semi-affine coordinates by the tuple $(U, Z_2V, Z_2, z_2 = Z_2^2)$ where Z_2 is a weight value. We first derive formulae for traditional operations of doubling and mixed addition. Then we improve the double-add formulae presented by Fan and Gong [14]. Finally, we obtain triplings to make use of the multibase representations introduced by Longa and Miri [24]. We obtain up to 11.78% speed up for scalar multiplication using binary expansions of scalars. Application of multibase methods using tripling formula derived in this work results up to 2.35% speed up compared to binary methods using formulae obtained in this work. Reader is referred to Section 5.5 for the environments considered for the comparison and for the analytical comparison.

5.1 Doubling

Our doubling formula, presented in Table A.1 in page 82 for the semi-affine coordinates costs one inversion, twenty multiplications and four squaring operations. In comparison to existing doubling formula in affine coordinates [22] due to Lange(2003), we save two multiplications and a squaring operation. We derive our formula from Harley's Algorithms presented in Algorithm 3 and explicit formula presented in Table 2.1. The input is a divisor class represented as $(U_1, \bar{V}_1 = Z_{12}V_1, Z_{12}, z_{12} = Z_{12}^2)$ where $D_1 = (U_1, V_1)$ is the corresponding reduced divisor. The output is a reduced divisor D_2 in the form of $(U_2, \bar{V}_2 = Z_{22}V_2, Z_{22}, z_{22} = Z_{22}^2)$ where $D_2 = 2D_1 = (U_2, V_2)$.

1. We compute $V' = 2\bar{V}_1$ in step 1 of Table A.1.

2. We compute $k_1x + k_0 = K \pmod{U_1}$ where $K = (z_{12}f - \bar{V}_1^2)/U_1$ in steps 2-3 of Table A.1.

   We obtain k_1 and k_0 as follows:

$$k_1 = (3u_{11}^2 + f_3 - 2u_{10})z_{12}$$

$$k_0 = (u_{11}(4u_{10} - f_3 - u_{11}^2) + f_2)z_{12} - v_{11}^2$$

3. We compute S'' the monic form of $S = s_1x + s_0$ and $Z_{22} = 1/s_1$ in steps 4-5 of Table A.1).

   We set up a linear system of two equations given by:

$$c(k_1x + k_0) = (x + s_0'')(2\bar{V}_1) - 2\bar{v}_{11}U_1$$

   on two variables c and s_0'' . For the computation of $S = K/2V_1 \pmod{U_1}$ we use $K \pmod{U_1}$ with weight z_{12} and the denominator with weight Z_{12}. Hence $c = 1/(Z_{12}s_1)$. We obtain d,$dc = d/(Z_{12}s_1)$ and ds_0'' at a cost of eight multiplications using Cramer's rule. If d and/or c is zero then we use the Cantor's algorithm given in Algorithm 1. We invert d to obtain $i = d^{-1}$. Using the inversion i of d, we compute $c = 1/(Z_{12}s_1) = (d/(Z_{12}s_1))i$ and $s_0'' = (ds_0'')i$ at a cost $I + 2M$. With an additional $1M$, We obtain $Z_{22} = 1/s_1$ from $1/(Z_{12}s_1)$. We compute $\bar{V}_1 = V_1/s_1 = (\bar{V}_1)(1/(Z_{12}s_1))$ at a cost of $2M$ in step 6 of Table A.1.

4. We perform computation of U_2 in step 7 of Table A.1.

   Using the expression given in Table 2.1 [22], we compute U_2 as:

$$u_{21} = 2s_0'' - z_{22}$$

$$u_{20} = s_0''^2 + 2\bar{v}_{11} + 2u_{11}z_{22}$$

   using $1M + 1S$.

5. We derive $\bar{V}_2 = Z_{22}V_2 = -S''U_1 - \bar{V}_1 \pmod{U_2}$ in step 8 of Table A.1)

The cost for this step is 3M.

$$\bar{V}_2 = -S''U_1 + S''U_2 + (u_{11} - u_{21})U_2 - \bar{V}_1$$

$$\bar{v}_{21} = (s_0'' - u_{21})(u_{21} - u_{11}) + u_{20} - u_{10} - \bar{v}_{11}$$

$$\bar{v}_{20} = s_0''(u_{20} - u_{10}) + (u_{11} - u_{21})(u_{20}) - \bar{v}_{10}$$

5.2 Mixed Addition

We derive an explicit formula for mixed addition on semi-affine coordinates using Harley's Algorithm for addition presented in Algorithm 2 and explicit formulae for addition given in Table 2.2. We are given two divisor classes D_1 and D_2 and we compute $D_3 = D_1 + D_2$. The inputs and the output are in the following form.

1. $(U_1, \bar{V}_1 = Z_{12}V_1, Z_{12}, z_{12} = Z_{12}^2)$ where $D_1 = (U_1, V_1)$ is a reduced divisor.
2. $D_2 = (U_2, V_2)$ is a reduced divisor.
3. $D_3 = (U_3, \bar{V}_3 = Z_{32}V_3, Z_{32}, z_{32} = Z_{32}^2)$ where $D_3 = D_1 + D_2 = (U_3, V_3)$ is a reduced divisor.

The cost of the new formula, shown in Table A.2 of page 83, is one inversion, twenty multiplications and a squaring operation. We save two multiplications and two squaring operations compared to existing addition in affine coordinates.

1. We normalize V_2 with respect to $\bar{V}_1$ in step 1 of Table A.2.
   By using two multiplications, we compute $\bar{V}_2 = Z_{12}V_2$.
2. We compute S'' the monic form of $S = s_1x + s_0$ and $Z_{22} = 1/s_1$ in step 2 of Table A.2.
   We set up a linear system of two equations given by:

   $$c(\bar{V}_2 - \bar{V}_1) = (x + s_0'')(U_1 - U_2) - (u_{11} - u_{12})U_2$$

   on two variables c and s_0'' . Since for the computation of $S = (V_2 - V_1)/U_1 \pmod{U_2}$ we use $(\bar{V}_2 - \bar{V}_1)$ with weight Z_{12} and the denominator with weight

1, it follows that $c = 1/(Z_{12}s_1)$. We obtain d,$dc = d/(Z_{12}s_1)$ and ds_0'' at a cost of eight multiplications using Cramer's rule. If d and/or c is zero then we use Cantor's algorithm (Algorithm 1). We use the inversion i of d to obtain $1/Z_{12}s_1 = (d/(Z_{12}s_1))i$ and $s_0'' = (ds_0'')i$ at a cost of $I + 2M$. With one multiplication, $Z_{32} = 1/s_1$ is derived from $1/(Z_{12}s_1)$. We also compute $z_{32} = Z_{32}^2$ using $1S$.

3. We obtain $\bar{V}_1 = V_1/s_1 = (\bar{V}_1)(1/(Z_{12}s_1))$ at a cost of $2M$ in step 3 of Table A.2.

4. For the computation of U_3 we use the expression given in Table 2.2 [22] with some optimization (Step 4 of Table A.2).

$$u_{31} = 2s_0'' + (u_{11} - u_{21}) - z_{32}$$

$$u_{30} = s_0''(s_0'' + 2a_{10}) + u_{10} - u_{20} - u_{21}(u_{11} - u_{12}) + 2\bar{v}_{11} + z_{22}(u_{11} + u_{12})$$

Since we have computed z_{22} and $u_{21}(u_{11}-u_{21})$ (when we computed linear system parameters), this step costs $2M$.

5. We obtain V_3 in step 5 of Table A.2. This step is similar to the corresponding step of the doubling formula. We perform the computation of $\bar{V}_3$ as follows:

$$\bar{V}_3 = -S''U_1 + S''U_3 + (u_{11} - u_{31})U_3 - \bar{V}_1$$

$$\bar{v}_{31} = (s_0'' - u_{31})(u_{31} - u_{11}) + u_{30} - u_{10} - \bar{v}_{11}$$

$$\bar{v}_{30} = s_0''(u_{30} - u_{10}) + (u_{11} - u_{31})(u_{30}) - \bar{v}_{10}$$

5.3 Double-Add Operation

We derive two formulae for double-add operation, one requiring two inversions and another requiring only one inversion. The second formula trades an inversion for a few multiplications in comparison to the first formula. The input divisors and the output divisors are represented as follows.

1. $(U_1, \bar{V}_1 = Z_{12}V_1, Z_{12}, z_{12} = Z_{12}^2)$ where $D_1 = (U_1, V_1)$ is a reduced divisor.

2. $D_2 = (U_2, V_2)$ is a reduced divisor.

3. $(U_4, \bar{V}_4 = Z_{42}V_4, Z_{42}, z_{42} = Z_{42}^2)$ where $D_4 = D_1 + D_2 = (U_4, V_4)$ is a reduced divisor.

5.3.1 Double-Add Operation with Two Inversions

We obtain a formula costing two inversions, thirty-six multiplications and two squaring operations. The formula is presented in Tables A.3 and A.4 of pages 84 and 85. Compared to the formula given in [14], we save six multiplications and three squaring operations.

1. We normalize V_2 with respect to $\bar{V}_1$ in step 1 of Table A.3.
   At a cost of two multiplications we compute $\bar{V}_2 = Z_{12}V_2$.

2. We compute S'' the monic form of $S = s_1x + s_0$ and $Z_{22} = 1/s_1$ in step 2 of Table A.3.
   We set up a linear system of two equations given by:

$$c(\bar{V}_2 - \bar{V}_1) = (x + s_0'')(U_1 - U_2) - (u_{11} - u_{12})U_2$$

   on two variables c and s_0'' . For the computation of $S = (V_2 - V_1)/U_1 \pmod{U_2}$ we use $(\bar{V}_2 - \bar{V}_1)$ with weight Z_{12} and the denominator with weight 1. Hence c is equal to $1/(s_1Z_{12})$. We obtain d,$dc = d/(Z_{12}s_1)$ and ds_0'' at a cost of eight multiplications using Cramer's rule. The inversion i of d is used to obtain $c = 1/(Z_{12}s_1) = (d/(s_1Z_{12}))i$ and $s_0'' = (ds_0'')i$ at a cost of $I + 2M$. With an additional $1M$, $Z_{22} = 1/s_1$ is computed from $1/(Z_{12}s_1)$. We also compute $z_{22} = Z_{22}^2$ using $1S$.

3. We adjust $\bar{V}_1 = V_1/s_1 = (\bar{V}_1)(1/(Z_{12}s_1))$ at a cost of $2M$ (Step 3 of Table A.3).

4. For the computation of U_3, we use the expression given in Table 2.2 [22] with some optimization (Step 4 of Table A.3):

$$u_{31} = 2s_0'' + (u_{11} - u_{21}) - z_{22}$$

$$u_{30} = s_0''(s_0'' + 2a_{10}) + a_{00} - u_{21}(u_{11} - u_{12}) + 2\bar{v}_{11} + z_{22}(u_{11} + u_{12})$$

Since we have computed z_{22} and $u_{21}(u_{11} - u_{21})$ (when we computed the linear system parameters), this step costs $2M$.

5. By applying Lemma 1, we compute the $\tilde{S}$ as (Step 5 of Table A.4):

$$(1/Z_{32})\tilde{S} \equiv -S'' - 2\bar{V}_1/U_1 \pmod{U_3}$$

We set up a linear system to find $\tilde{T} = x + \tilde{t}_0''$ the monic form of $(-2\bar{V}_1/U_1) \pmod{U_3}$ as

$$-\tilde{c}'2\bar{V}_1 = (x + \tilde{t}_0'')(U_1 - U_3) - (u_{11} - u_{31})U_{31}$$

and solve to obtain $\tilde{d}', \tilde{c}'' = \tilde{d}'\tilde{c}'$ and $\tilde{t}_0''' = \tilde{d}'\tilde{t}_0''$. If $\tilde{d}'\tilde{c}'$ is zero, then we apply Cantor's algorithm. Otherwise we proceed to perform normalized addition:

$$\tilde{S} = -\tilde{c}''S'' + \tilde{T}'''(= \tilde{d}'X + \tilde{t}_0''')$$

$$\tilde{d} = \tilde{d}' - \tilde{c}''$$

$$\tilde{d}\tilde{c} = \tilde{c}''$$

$$\tilde{d}\tilde{s}'' = \tilde{t}_0''' - \tilde{c}'s_0''$$

Next step is to invert $\tilde{d}$ and obtain $\tilde{c}$ and $\tilde{s}_0''$. We also compute $Z_{42} = d\tilde{c}Z_{32}$ and $z_{42} = Z_{42}^2$. The cost of this step is one inversion, twelve multiplications and a squaring operation.

6. After adjusting $\bar{V}_1$ by : $\tilde{V}_1 = V_1/s_1 = \bar{V}_1\tilde{c}$, we compute U_4 and V_4 as follows (Steps 6-8 of Table A.4):

$$u_{41} = 2\tilde{s}_0'' + (u_{11} - u_{31}) - z_{42}$$

$$u_{40} = \tilde{s}_0''(\tilde{s}_0'' + 2(u_{31} - u_{11})) + u_{10} - u_{30} - u_{31}(u_{11} - u_{31}) + 2\bar{v}_{11} + z_{42}(u_{11} + u_{31})$$

$$\bar{V}_4 = -\tilde{S}''U_1 + \tilde{S}''U_4 + (u_{11} - u_{41})U_4 - \tilde{V}_1$$

$$\bar{v}_{41} = (\tilde{s}_0'' - u_{41})(u_{41} - u_{11}) + u_{40} - u_{10} - \tilde{v}_{11}$$

$$\bar{v}_{40} = \tilde{s}_0''(u_{40} - u_{10}) + (u_{11} - u_{41})(u_{40}) - \tilde{v}_{10}$$

5.3.2 Double-Add Operation with One Inversion

We derive a formula costing one inversion, forty-five multiplications and four squaring operations. The formula is shown in Tables A.5 and A.6 of pages 86 and 87. Compared to the formula given in [14], we save eleven multiplications and three squaring operations.

1. We normalize V_2 with respect to $\bar{V}_1$ in step 1 of Tables A.5 and A.6.
   At the expense of two multiplications we compute $\bar{V}_2 = Z_{12}V_2$.

2. We obtain S'', the monic form of $S = s_1x + s_0$, and $Z_{22} = 1/s_1$ in step 2 of Tables A.5 and A.6.
   We set up a linear system of two equations given by:

   $$c(\bar{V}_2 - \bar{V}_1) = (x + s_0'')(U_1 - U_2) - (u_{11} - u_{12})U_2$$

   on two variables c and s_0''. c is equal to $c_1/(Z_{12}s_1)$, because for the computation of $S = (V_2 - V_1)/U_1 \pmod{U_2}$ we use $(\bar{V}_2 - \bar{V}_1)$ with weight Z_{12} and the denominator with weight 1. We obtain d,$c' = d/(Z_{12}s_1)$ and $s_0''' = ds_0''$ at a cost of eight multiplications using Cramer's rule. In addition we compute $d' = dc'$,$Z_{32} = c'Z_{12}$ and $z_{32} = Z_{32}^2$.

3. We adjust $\bar{V}_1 = (d^2V_1)/s_1 = d'\bar{V}_1$ at a cost of $2M$ (Step 3 of Tables A.4 and A.5).

4. We obtain $D = d^2$ and DU_{21} in step 4 of Tables A.5 and A.6.

   $$D = d^2, \quad , u_{21}' = Du_{21}, \quad u_{20}' = Du_{20}$$

   The cost for this step is two multiplications and a squaring operation.

5. Instead of U_3 we compute $D(U_3 - U_1) = u'_{31}x + u'_{30}$ where $D = d^2$ in step 5 of Tables A.5 and A.6.

$$W_1 = ds_0^{'''}, u'_{31} = 2W_1 - u'_{21} - z_{32}$$

$$u'_{30} = s_0^{'''2} + (2W_1 - u'_{21})(u_{11} - u_{21}) - u'_{20} + +2\bar{v}_{11} + z_{32}(u_{11} + u_{21})$$

We use two multiplications and a squaring operation.

6. By applying Lemma 1, we compute the $\tilde{S}$ as (Steps 6-9 of Tables A.5 and A.6):

$$(1/Z_{32})\tilde{S} \equiv -dS^{'''} - 2\bar{V}_1/U_1 \pmod{U_3}$$

First we compute $(V_1 - V_3)/U_3 \pmod{U_1}$. We observe that:

$$(V_1 - V_3)/U_3 \equiv S + u_{11} - u_{31} - 2V_1/U_3 \pmod{U_1}$$

from the fact that:

$$V_3 = -SU_1 + SU_3 + (u_{11} - u_{31})U_3 - V_1$$

We set up a linear system to find $\tilde{T} = \tilde{t}_1 x + \tilde{t}_0 \equiv -2\bar{V}_1/U_3 \pmod{U_1}$ as

$$-2\bar{V}_1 = (\tilde{t}_1 x + \tilde{t}_0)(U_3 - U_1) + \tilde{t}'_1(u_{11} - u_{31})U_{31}$$

and by applying Cramer's rule, we obtain $\tilde{d}', \tilde{t}'_1 = \tilde{d}'\tilde{t}_1$ and $\tilde{t}'_0 = \tilde{d}'\tilde{t}_0$. By performing normalized addition of $(S + u_{11} - u_{31})$ and $T' = t'_1 x + t'_0$, we obtain

$$\tilde{d} = D(\tilde{t}'_1 + \tilde{d}')$$

$$i_1 = \tilde{d}^{-1}$$

$$\tilde{s}''_0 = (Dt'_0 + (W_1 - u'_{31})\tilde{d}')i_1$$

$$\tilde{c} = \tilde{d}'i_1$$

$$Z_{42} = d\tilde{c}Z_{32}, z_{42} = Z_{42}^2$$

Then we adjust s_0'', U_3 and V_1:

$$i_2 = i_1(\tilde{t}_1' + \tilde{d}')$$

$$u_{31} = u_{31}' i_2 + u_{11}$$

$$u_{30} = u_{30}' i_2 + u_{10}$$

$$\tilde{s}_0'' = \tilde{s}_0'' + u_{31} - u_{11}$$

$$\tilde{v}_{11}' = \bar{v}_{11}\tilde{c}$$

$$\tilde{v}_{10}' = \bar{v}_{10}\tilde{c}$$

7. We derive $U_4 = X^2 + u_{41}x + u_{40}$ in step 10 of Tables A.5 and A.6.

$$u_{40} = \tilde{s}_0''^2 + (2\tilde{s}_0'' + u_{31})(u_{11} - u_{31}) + u_{30} - u_{10} + 2\tilde{v}_{11} + (u_{11} + u_{31})z_{42}$$

$$u_{41} = 2s_0'' + u_{31} - u_{11} - z_{32}$$

8. Finally, we compute $\bar{V}_4 = Z_{42}V_4 = \bar{v}_{41}X + \bar{v}_{40}$ in step 11 of Tables A.5 and A.6.

$$\bar{v}_{41} = (u_{41} - \tilde{s}_0'')(u_{11} - u_{41}) + (u_{40} - u_{10}) - \tilde{v}_{11}$$

$$\bar{v}_{40} = \tilde{s}_0''(u_{40} - u_{10}) + (u_{11} - u_{31})u_{40} - \tilde{v}_{10}$$

5.4 Tripling

Tripling formulae are based on the fact that the input divisor and the output divisor are represented in the semi-affine coordinates. We derive two tripling formulae. The first formula as in the case of double-add operation requires two inversions. The second formula is obtained by trading one inversion for other field operations.

5.4.1 Tripling with Two Inversions

We perform tripling with two inversions as doubling followed by addition. Using Lemma 1, we avoid computing intermediate V as in the case of double-add operation. The formula is presented in Tables A.7 and A.8 of pages 88 and 89.

1. We compute $V' = 2\bar{V}_1$ in step 1 of Table A.7.

2. We obtain $k_1 x + k_0 = K z_{12} \pmod{U_1}$ in steps 2-3 of A.7.
   For the doubling part, as before we use the following expression given in [22].

$$k_1 = 3u_{11}^2 + f_3 - 2u_{10}$$

$$k_0 = u_{11}(4u_{10} - f_3 - u_{11}^2) + f_2 - v_{11}^2$$

   Since exact v_{11} is not given the expressions have to be normalized. In our formula we compute as $K \pmod{U_1}$,

$$k_1 = (3u_{11}^2 + f_3 - 2u_{10})z_{12}$$

$$k_0 = (u_{11}(4u_{10} - f_3 - u_{11}^2) + f_2)z_{12} - v_{11}^2$$

   A weighted form of $K \pmod{U_1}$ is computed at a cost of three multiplications and two squaring operations. The cost is $3M + 2S$.

3. We compute the monic form S'' of $S = s_1 x + s_0$ and $Z_{22} = 1/s_1$ in steps 4-5 of Table A.7.
   Now we set up a linear system

$$c(k_1 x + k_0) = (x + s_0'')(2\bar{V}_1) - 2\bar{v}_{11}U_1$$

   on two variables c and s_0''. Similar to the computation of doubling, we obtain Z_{22}, z_{22}, s_0'' at a cost of $I + 11M + 1S$ operations using Cramer's rule.

4. We adjust $\bar{V}_1$ at a cost of two multiplications in step 6 of Table A.7.

$$\bar{v}_{11} = c\bar{v}_{11}, \bar{v}_{10} = c\bar{v}_{10}$$

5. By using the expression given in Table 2.1 [22], we compute U_2 as (Step 7 of A.7):

$$u_{21} = 2s_0^{''} - z_{22}$$

$$u_{20} = s_0^{''2} + 2\bar{v}_{11} + 2u_{11}z_{22}$$

The cost is $1M + 1S$.

6. By applying Lemma 1, we compute the $\tilde{S}$ as (Step 8 of Table A.8):

$$(1/Z_{32})\tilde{S} \equiv -S^{''} - 2\bar{V}_1/U_1 \pmod{U_3}$$

We set up a linear system to find the monic form $\tilde{T} = x + \tilde{t}_0''$ of $-2\bar{V}_1/U_1 \pmod{U_3}$

$$-\tilde{c}'2\bar{V}_1 = (x + \tilde{t}_0^{''})(U_1 - U_3) - (u_{11} - u_{31})U_{31}$$

and solve to obtain $\tilde{d}', \tilde{c}''\tilde{d}'\tilde{c}'$ and $\tilde{t}_0''' = \tilde{d}'\tilde{t}_0''$. If $\tilde{c}''$ is zero, then we apply Cantor's algorithm. Otherwise we proceed to perform normalized addition:

$$\tilde{S} = -\tilde{c}''S^{''} + \tilde{T}''$$

$$\tilde{d} = \tilde{d}' - \tilde{c}''$$

$$\tilde{d}\tilde{c} = \tilde{c}''$$

$$\tilde{d}\tilde{s}'' = \tilde{t}_0''' - \tilde{c}''s_0''$$

Next step is to invert $\tilde{d}$ and obtain $\tilde{c}$ and $\tilde{s}_0''$. We also compute $Z_{32} = \tilde{c}Z_{22}$ and $z_{32} = Z_{32}^2$. The cost of this step is one inversion, twelve multiplications and a squaring operation.

7. After adjusting $\bar{V}_1$ as $\tilde{V}_1 = V_1/s_1 = (\bar{V}_1)(\tilde{c})$, we compute U_3 and V_3 by performing

the following steps (Steps 9-11 of Table A.8).:

$$u_{31} = 2\tilde{s}_0'' + (u_{11} - u_{21}) - z_{32}$$

$$u_{30} = \tilde{s}_0''(\tilde{s}_0'' + 2(u_{21} - u_{11})) + u_{10} - u_{20} + u_{21}(u_{11} - u_{21}) + 2\tilde{v}_{11} + z_{32}(u_{11} + u_{21})$$

$$\bar{V}_3 = -\tilde{S}''U_1 + \tilde{S}''U_3 + (u_{11} - u_{31})U_3 - \tilde{V}_1$$

$$\bar{v}_{31} = (\tilde{s}_0'' - u_{31})(u_{31} - u_{11}) + u_{30} - u_{10} - \tilde{v}_{11}$$

$$\bar{v}_{30} = \tilde{s}_0''(u_{30} - u_{10}) + (u_{11} - u_{31})(u_{30}) - \tilde{v}_{10}$$

5.4.2 Tripling with One Inversion

Tripling with one inversion is obtained by performing doubling followed by addition and while postponing the inversion needed in the doubling step of the tripling to addition step. The formula is presented in Tables A.9 and A.10 of pages 90 and 91.

1. We compute $V' = 2\bar{V}_{11}$ in step 1 of Table A.9.

2. We obtain $k_1 x + k_0 = z_{12}K \pmod{U_1}$ in steps 2-3 of Table A.9.
   For the doubling part, as before we use the following expression given in [22].

$$k_1' = 3u_{11}^2 + f_3 - 2u_{10}$$

$$k_0' = u_{11}(4u_{10} - f_3 - u_{11}^2) + f_2 - v_{11}^2$$

   Since exact v_{11} is not given, the expressions have to be normalized. We compute K as,

$$k_1 = (3u_{11}^2 + f_3 - 2u_{10})z_{12}$$

$$k_0 = (u_{11}(4u_{10} - f_3 - u_{11}^2) + f_2)z_{12} - v_{11}^2$$

   A weighted form of $K \pmod{U_1}$ is computed at a cost of three multiplications and two squaring operations $(3M + 2S)$.

3. Now we set up a linear system (Steps 4-5 of Table A.9)

$$c(k_1 x + k_0) = (x + s_0'')(2\bar{V}_1) - 2\bar{v}_{11}U_1$$

on two variables c and s_0''. For the computation of $S = (K/2V_1) \pmod{U_2}$ we use $K \pmod{U_1}$ with weight z_{12} and the denominator with weight Z_{12}. Hence c is equal to $1/(Z_{12}s_1)$. We obtain d,$c' = d/(Z_{12}s_1)$ and $s_0''' = ds_0''$ at a cost of eight multiplications using Cramer's rule. With additional $1M + 1S$, $Z_{22} = c'Z_{12} = d/s_1$ and $z_{22} = Z_{22}^2$ are computed.

4. We adjust $\bar{V}_1 = (d^2V_1)/s_1 = (d\bar{V}_1)(1/(Z_{12}s_1))$ at a cost of $3M$ (Step 6 of Table A.9).

5. Using the expression given in [22], we compute DU_2 and DU_1 as (Step 7-8 of Table A.9):

$$W_6 = ds_0''', u_{21} = 2W_0 - z_{22}$$

$$u_{20} = s_0''''^2 + 2\bar{V}_1 + 2u_{11}z_{22}$$

$$D = d^2$$

$$u_{11}' = Du_{11}$$

$$u_{10}' = Du_{10}$$

6. By the application of Lemma 1, we compute the $\tilde{S}$ as (Step 9 of Table A.10):

$$(1/Z_{22})\tilde{S} \equiv (-dS''' - 2(\bar{V}_1/U_1)) \pmod{U_3}$$

We set up a linear system to find $\tilde{T} = \tilde{t}_1 x + \tilde{t}_0 \equiv -2\bar{V}_1/U_1 \pmod{U_3}$ as

$$-2\bar{V}_1 = (\tilde{t}_1 x + \tilde{t}_0)(U_1 - U_2) - (u_{11} - u_{21})U_{21}$$

and solve to obtain $\tilde{d}', \tilde{t}' = \tilde{d}'\tilde{t}_1$ and $\tilde{t}_0' = \tilde{d}'\tilde{t}_0$. Then by performing normalized

addition of $-S$ and $T' = t'_1 x + t'_0$, we obtain:

$$\tilde{d} = D(\tilde{t}'_1 - \tilde{d}')$$

$$i_1 = \tilde{d}^{-1}$$

$$\tilde{s}''_0 = (Dt'_0 - (W_6)\tilde{d}')i_1$$

$$\tilde{c} = \tilde{d}' i_1 = \tilde{d}/(\tilde{s}_1 Z_{22})$$

$$Z_{32} = d\tilde{c}Z_{22}, z_{32} = Z_{32}^2$$

At this stage, we adjust U_2 and V_1 (Steps 10-11 of Table A.10):

$$i_2 = i_1(\tilde{t}'_1 - \tilde{d}')$$

$$u_{21} = u'_{21} i_2$$

$$u_{20} = u'_{20} i_2$$

$$\tilde{v}'_{11} = \tilde{c}v'_{11}$$

$$\tilde{v}'_{10} = \tilde{c}v'_{10}$$

7. We compute U_3 and V_3 as follows (Steps 12-13 of Table A.10):

$$u_{31} = 2\tilde{s}''_0 + (u_{11} - u_{21}) - z_{32}$$

$$u_{30} = \tilde{s}''^2_0 + (2\tilde{s}''_0 + u_{21})(u_{21} - u_{11}) + u_{10} - u_{20} + 2\tilde{v}_{11} + z_{32}(u_{11} + u_{21})$$

$$\bar{V}_3 = -\tilde{S}'' U_1 + \tilde{S}'' U_3 + (u_{11} - u_{31})U_3 - \tilde{V}_1$$

$$\bar{v}_{31} = (\tilde{s}''_0 - u_{31})(u_{31} - u_{11}) + u_{30} - u_{10} - \tilde{v}_{11}$$

$$\bar{v}_{30} = \tilde{s}''_0(u_{30} - u_{10}) + (u_{11} - u_{31})(u_{30}) - \tilde{v}_{10}$$

5.5 Precomputation Schemes and the Cost of Scalar Multiplication Using the New Formulae

5.5.1 Comparison of Costs of Formulae

Table 5.1 compares costs of formulae for semi-affine coordinates with existing formulae. We compare doubling and mixed addition with the cost given in [22]. Double-add operation cost is compared with the formulae given in [14]. Tripling cost is compared with naive method of doubling followed by addition.

5.5.2 Precomputation

Cost of scalar multiplication depends on the number representation being using and the costs of field operations in the environment concerned. We illustrate the improvements using the $mp\mathbb{F}_q$ library [16]. The $mp\mathbb{F}_q$ library has implementations of general fields and implementations that are optimized for some particular fields. We make our comparison on the prime fields with two words long characteristic. In that implementation the cost of an inversion is 16.7 multiplications and that of a squaring operation is 0.9 multiplications. The cost of precomputation using existing methods is one (group) doubling and $2^{w-2}-1$ (group) additions. We use the double-add and tripling operation to decrease the computational cost of precomputation. We illustrate our technique for the cases with $w=3$ and $w=4$. We denote by D the base point to be multiplied by the scalar. For $w=3$, we need to precompute $3D$. For $w=4$ we need to precompute $3D, 5D$ and $7D$.

Only one precomputed point ($w=3$)

We modify step 9 of tripling with one inversion shown on Tables A.7 and A.8 as follows:

$$\tilde{a}_{00} = u'_{10} - u_{20}, \tilde{a}_{10} = u'_{11} - u_{21}, \tilde{a}_{01} = -\tilde{a}_{10}u_{20}$$

$$\tilde{W}_0 = \tilde{a}_{10}u_{21}, \tilde{a}_{11} = \tilde{a}_{00}D + \tilde{W}_0, \tilde{q}_0 = 2\bar{v}_{10}, \tilde{q}_1 = 2\bar{v}_{11}$$

$$\tilde{c} = (\tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{10}\tilde{a}_{01}), \tilde{W}_1 = (\tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_1)d - \tilde{d}', \tilde{d} = d\tilde{W}_1$$

$$\tilde{s}_0''' = (\tilde{q}_0\tilde{a}_{11} - \tilde{q}_1\tilde{a}_{01})d - \tilde{d}'s_0'''$$

$$M_1 = \tilde{c}Z_{22}, I = (\tilde{d}M_1)^{-1}$$

$$\tilde{s}_1 = I\tilde{d}'^2$$

$$i_1 = IM_1$$

$$\tilde{s}_0'' = \tilde{s}_0'''i_1, \tilde{c} = \tilde{c}i_1$$

$$Z_{32} = \tilde{c}Z_{22}, z_{32} = Z_{32}^2$$

That is, we use Montgomery simultaneous inversions to obtain $\tilde{s}_1$ at an additional cost of $4M + 2S$. Unweighted V_3 can be obtained with two multiplication.

Three Precomputed Points($w = 4$)

In this case, we have to compute, $3D, 5D$ and $7D$. We proceed as follows:

1. We compute $3D$ using tripling introduced earlier.

2. Note that we did not explicitly compute the second polynomial V_2 of the Mumford representation of $2D$. To obtain V_2 we note that:

$$\bar{V}_3 = -S''U_1 + \bar{S}''U_3 - (V_1/s_1)$$

where $\bar{S} = S + u_{11} - u_{31}$. Almost always U_2 is degree 2 and hence, for the frequent case, V_3 remains unchanged when modular reduced by U_2. But by Harley's algorithm:

$$S''U_1 = (V_2 - V_1)/s_1 \pmod{U}_2$$

Hence we obtain the following expression for V_2:

$$\bar{V}_2 = V_2/s_1 = -V_3 + \bar{S}''U_3 \pmod{U_2}$$

$$\bar{v}_{31} = (-\tilde{s}_0'' + u_{41} - u_{11} - u_{31})(u_{31} - u_{41}) + u_{40} - u_{30} - \tilde{v}_{41}$$

$$\bar{v}_{30} = (u_{11} - u_{41} + \tilde{s}_0'')(u_{40} - u_{30}) + (u_{31} - u_{41})(u_{30}) - \tilde{v}_{40}$$

We obtain $\bar{V}_2$ at a cost of three multiplications.

3. By noting that $\bar{V}_2$ and $\bar{V}_3$ both have the same weight, we perform a double-add operation to obtain $7D = 2(2D) + 3D$, side-stepping the first step of normalization. We use $7M+2S$ computation to find $1/Z_{32}$ and $1/Z_{72}$. $4M+2S$ operations are used to obtain the inverse of the weight of Z_{72} similar to what was done for the case of one precomputed point. Performing Montgomery simultaneous inversions to obtain the inverse of the weight of V_3 costs three multiplications.

4. Six multiplications are used to obtain $3D, 5D$ and $7D$ in affine form.

5. Two multiplications are saved on normalization.

Total cost is one tripling, one double-add and $17M + 1S$.

5.5.3 Cost of Main Phase and Total Cost

The Tables 5.3 and 5.4 show the speed up obtained for the main phase of scalar multiplication using double-add operation with two inversions and one inversion respectively using binary expansions of scalars. Table 5.5 shows the total cost of scalar multiplication for a 256-bit scalar using binary expansions of scalars using double-add operation with one inversion. Table 5.6 shows the total cost of scalar multiplication using $wmbNAF$ with bases 2 and 3 against weights on base 2. The weight on base 3 is zero. We use the double-add operation and tripling with one inversion. The salient features of improvements are as follows:

1. Up to 11.78% improvement in the total cost using binary expansions of scalars.

2. Total cost is improved by up to 2.35% to using mutibase methods compared to binary expansion methods using our formulae

Table 5.1: Costs of Formulae for Semi-Affine Coordinates

Formulae	**This Work**	**Existing**
Doubling	I+20M+4S	I+22M+5S
Addition	I+20M+1S	I+22M+3S
Double-Add	2I+36M+2S	2I+42M+5S
Tripling	2I+36M+5S	2I+38M+5S
Double-Add	I+45M+4S	I+56M+7S
Tripling	I+47M+6S	-

Table 5.2: Costs of Precomputations for Semi-Affine Coordinates

Window	**Existing**	**This work**	**Reduction**
3	84.6M	74.9M	11.46%
4	167.4M	153.2M	8.48%

Table 5.3: Costs per Bit Scalar of main phase of scalar multiplication using double-add with two inversions for Semi-Affine Coordinates

Window	**Existing**	**This work**	**Reduction**
2	(4I+86M+15S)/3	(4I+76M+10S)/3	8.71%
3	(5I+108M+20S)/4	(5I+96M+14S)/4	8.30%
4	(6I+130M+25S)/5	(6I+116M+18S)/5	8.03%

Table 5.4: Costs per Bit Scalar of main phase of scalar multiplication using double-add with one inversion for Semi-Affine Coordinates

Window	**Existing**	**This work**	**Reduction**
2	(3I+100M+17S)/3	(3I+85M+12S)/3	11.78%
3	(4I+122M+22S)/4	(4I+105M+16S)/4	10.74%
4	(5I+144M+27S)/5	(5I+125M+20S)/5	10.05%

Table 5.5: Costs for Scalar Multiplications using double-add with one inversion for 256-Bit Scalar in Semi-Affine Coordinates using Binary Methods

Window	**Existing**	**This work**	**Reduction**
3	13435M	11992M	10.74%
4	13059M	11750M	10.02%

Table 5.6: Costs for Scalar Multiplications using double-add and tripling with one inversion for 256-Bit Scalar in Affine Coordinates Using $(2,3) - wmbNAF_w$ Methods

Weight of 2 (w)	**Binary**	$(2,3) - wmbNAF_w$	**Reduction**
2	12450M	12157M	2.35%
3	11992M	11846M	1.21%

5.6 Summary

In this chapter we obtained efficient traditional and composite operations for environments in which inversion is cheap. Often inversions are very expensive compared to multiplications for low genus hyperelliptic curves over prime fields. Further, inversions on prime fields are difficult to implement in hardware. In the next chapter we improve point formulae for inversion-free coordinates.

Chapter 6

Formulae in Inversion-Free Coordinates

In the previous chapter, we presented formulae that require inversions. Field inversions are performed using the extended-Euclidean Algorithm for computing greatest common divisors. The extended Euclidean algorithm has high computational cost and is hard to implement in hardware. Inversion-free methods are the preferred methods for scalar multiplication on the Jacobian of low genus hyperelliptic curves over prime fields. We derive formulae in the two inversion-free coordinate systems that are being currently used for hyperelliptic curves using techniques developed in Chapter 4. They are Lange's new coordinates and projective coordinates. We obtain more efficient formulae for doubling operations and additions. We also introduce formulae for double-add and tripling operations. Lange's new coordinates need storage of four weight values in comparison to one value needed for projective coordinates. In environments in which memory is a major issue projective coordinates are useful. We reduce the gap between the cost of scalar multiplication for projective and Lange's new coordinates.

6.1 Doubling

Doubling formula for Lange's coordinates costs thirty multiplication and seven squaring operations. The formula is presented in Tables A.11 of page 92. We save seven multiplications at the expense of three squaring operations compared to the formula presented in [22]. For the projective coordinates our formula, presented in Table A.17 of page 98, costs thirty-five multiplications and two squaring operations. The cost of our formula is three multiplications and four squaring operations less than that was obtained in [22].

6.1.1 Lange's New Coordinates

When given a divisor D_1 as $(U_1, V_1, Z_{11}, z_{11}, Z_{12}, z_{12})$ in Lange's new coordinates, we proceed as follows to obtain $D_2 = 2D_1$ as $(U_2, V_2, Z_{21}, z_{21}, Z_{22}, z_{22})$ in Lange's new coordinates.

1. We compute $V_1' = 2V_1$ in step 1 of Table A.11.

2. We obtain of $k_1 x + k_0 \equiv K \pmod{U_1}$ in step 2 of Table A.11. We compute k_1 with weight $z_{11}^2 z_{12}$ and k_0 with weight $z_{11}^3 z_{12}$ as:

$$w_0 = v_{11}^2$$

$$w_1 = u_{11}^2$$

$$W_2 = z_{11}^2$$

$$W_3 = u_{10} z_{11}$$

$$W_4 = f_3 W_2 + w_1$$

$$w_5 = 2W_3$$

$$k_1 = (2w_1 + w_4 - w_5) z_{12}$$

$$k_0 = (u_{11}(2w_5 - w_4) + f_2 W_2 z_{11}) z_{12} - w_0$$

   In further discussion we denote by k_1' the value $k_1 z_{11}$ which is not computed in the formula.

3. We set up the linear system for computation $K/2V_1 \pmod{U_1}$ using the polynomial equation (Step 3 of Table A.11)

$$(k_1' x + k_0)c = (z_{11} x + s_0''')(v_{11}' x + v_{10}') - 2v_{11}'(z_{11} x^2 + u_{11} x + u_{10})$$

   We note that $s_0''' = z_{11} s_0''$. By equating coefficients of powers of x, we obtain the following linear system.

$$a_{00} = 2v_{10}; a_{01} = -k_0; q_0 = 2v_{11} u_{10};$$

$$a_{10} = 2v_{11}; a_{11} = -k_1'; q_1 = 2v_{11}u_{11} - 2v_0z_{11}$$

Instead of computing a_{00}, a_{11}, q_0 as above, we compute them using expressions given below.

$$a_{00} = v_{10}'z_{11}; a_{11} = -k_1; q_0 = v_{11}'W_3$$

We compute q_1 using the expression $v_{11}'u_{11} - a_{00}$. The determinant of the linear system is computed as follows.

$$d = a_{00}a_{11} - a_{01}a_{10}$$

We obtain $c' = z_{11}dc$ and $s_0'''' = ds_0'''$ by performing the following calculations.

$$c' = a_{00}q_1 - 2v_{11}q_0$$

$$s_0'''' = q_0a_{11} - q_1a_{01}$$

If $c' = 0$ then $\gcd(U_1, V_1)$ is not zero. If the determinant d is zero then s_1 is zero. These cases occur rarely. These cases can be handled by Cantor's algorithm without significant performance penalty on average. We do not consider these cases in our discussions. We observe that :

$$c' = dz_{11}c = (dz_{11}(z_{11})(1/(Z_{11}^3Z_{12}))/s_1 = (dZ_{11})/(Z_{12}s_1)$$

The above relation holds from the following facts.

(a) A weight of $1/Z_{11}^3Z_{12}$ is introduced by the differences in weights of denominator and numerator in the computation S.

(b) A weight of z_{11} is introduced by the fact that z_{11} is the leading coefficient pseudo-inverse in the linear system.

By similar arguments the following relationship holds between s_0'''' and its corresponding affine value s_0''.

$$s_0'''' = ds_0''' = dz_{11}s_0''$$

4. We perform some precomputation in step 4 of Table A.11.

$$W_6 = c' Z_{12}; W_7 = W_6 Z_{11}$$

$$W_8 = dz_{11}; W_9 = W_8^2$$

$$W_{10} = W_7^2; W_{11} = W_8 s_0'''$$

We note that $W_6 = (dZ_{11})/s_1$ and $W_7 = (dz_{11})/s_1$.

5. We adjust V_1 in step 5 of Table A.11. We compute $V_1' = (W_9 V_1)/s_1$ as shown below.

$$W_{12} = dc'; v_{11}' = W_{12} v_{11}; v_{10}' = W_{12} v_{10}$$

We obtain V' with a weight of W_9 with respect to its affine value.

6. We compute U_2 with weight W_9 in step 6 of Table A.11.

$$u_{21} = 2W_{11} - W_{10} =$$

$$u_{20} = s_0''''^2 + 2v_{11}' + 2u_{11} W_6^2$$

7. We compute a form of U_1 with weight as that of U_2 (Step 7 of Table A.11).

$$W_{13} = W_8 d; u_{11}' = W_{13} u_{11}; u_{10}' = W_{13} u_{10}$$

8. We set the weights in step 8 of Table A.11.

$$Z_{21} = W_8$$

$$z_{21} = W_9$$

$$Z_{22} = W_7$$

$$z_{22} = W_{10}$$

9. We compute V_2 in step 9 of Table A.11. $V_2 = -(z_{21}x + W_{11})U_1' - z_{21}V_1' \pmod{U_2}$.

V_2 has a weight of $z_{21}^2/s_1 = Z_{21}^3 Z_{22}$. V_2 is obtained as:

$$v_{20} = W_{11}(u_{20} - u'_{10}) + (u'_{11} - u_{21})u_{20} - z_{21}v'_{10}$$

$$v_{21} = (u_{21} - W_{11})(u'_{11} - u_{21}) + z_{21}(u_{20} - u'_{10} - v'_{10})$$

6.1.2 Projective Coordinates

When given a divisor D_1 as (U_1, V_1, Z_1) in projective coordinates, we proceed as shown subsequently to obtain $D_2 = 2D_1$ as (U_2, V_2, Z_2) in projective coordinates.

1. We compute $V_1' = 2V_1$ in step 1 of Table A.17.

2. We compute a weighted $k_1 x + k_0 \equiv K \pmod{U_1}$ in step 2 of Table A.17. Similar to the case of Lange's new coordinates, we compute k_1 and k_0 with different weights. We compute k_1 with weight Z_1^2 and k_0 with weight Z_1^3 as

$$w_0 = v_{11}^2$$

$$w_1 = u_{11}^2$$

$$W_2 = Z_1^2$$

$$w_3 = f_2 W_2$$

$$W_4 = u_{10} Z_1$$

$$w_5 = f_3 W_2 + w_1$$

$$w_6 = 2W_4$$

$$k_1 = 2w_1 + w_3 - w_5$$

$$k_0 = u_{11}(2w_5 - w_3) + (w_3 - w_0)Z_1$$

   In further discussion we denote by k_1' the value $k_1 Z_1$ which is not computed in the formula.

3. We set up the linear system for computation of $K/2V_1 \pmod{U_1}$ using the

polynomial equation presented below (Step 3 of the Table A.17).

$$(k_1'x + k_0)c = (Z_1x + s_0''')(2v_{11}x + 2v_{10}) - 2v_{11}(Z_1x^2 + u_{11}x + u_{10})$$

By equating the coefficients of powers of the indeterminate, we obtain two linear equations.

$$a_{00} = v_{10}'; a_{01} = -k_0; q_0 = v_{11}'u_{10};$$

$$a_{10} = v_{11}'; a_{11} = -k_1'; q_1 = v_{11}'u_{11} - v_0'Z_1$$

Instead of computing a_{00}, a_{11}, q_0 as above, we compute them as follows:

$$W_7 := v_{10}Z_1; a_{00} = 2W_7; a_{11} = -k_1; q_0 = v_{11}'W_4; a_{01} = -k_0; a_{10} = v_{11}'$$

The q_1 can be computed as $v_{11}'u_{11} - a_{00}$. The determinant of the linear system is computed as follows:

$$d = a_{00}a_{11} - a_{01}a_{10}$$

We obtain $c' = dZ_1c$ and $s_0'''' = Z_1s_0'''$ by the expressions given below.

$$c'' = a_{00}q_1 - a_{10}q_0 (= d/s_1)$$

$$s_0'''' = q_0a_{11} - q_1a_{01} (= dZ_1s_0'')$$

If $c' = 0$ then $\gcd(U_1, V_1)$ is not zero. As mentioned previously, the infrequent cases of $d = 0$ and/or $c' = 0$ are not considered in this discussion.

4. We precompute the following values (Step 4 of Table A.17).

$$W_8 = ds_0''''$$

$$w_9 = c'Z_1$$

$$W_{10} = c'w_9$$

$$W_{11} = dw_9$$

5. We adjust V_1 in step 5 of Table A.17.

$$v'_{11} = W_{11}v_{11}$$

$$v'_{10} = W_{11}a_{00}$$

Hence now v'_{11} has a weight $d^2Z_1^2$ w.r.t the corresponding affine value. Since $a_{00} = v_{10}Z_1$, v'_{10} has a weight $d^2Z_1^3$ w.r.t the corresponding affine value.

6. We compute U'_2 in step 6 of Table A.17.

$$u'_{21} = 2W_8 - W_{10}$$

$$u'_{20} = 2u_{11}W_{10} + 2v'_{11} + s_0^{''''2}$$

In this step we have computed u_{21} with a weight of d^2Z_1 and u_{20} with a weight of $(dZ_1)^2$.

7. We obtain weight Z_2 in step 7 of Table A.17.

$$W_{12} = d^2; W_{13} = W_{11}W_2; Z_2 = W_{12}W_{13}$$

8. We adjust U_1 in step 8 of A.17. In this step we compute u_{10} and u_{11} with weights as that of u_{20} and u_{21} respectively.

$$u'_{11} = W_{12}u_{11}; u'_{10} = W_{12}W_4$$

9. We compute of V_2 in step 9 of Table A.17.

$$v_{20} = W_8(u_{20} - u'_{10}) + (u'_{11} - u_{21})u_{20} - Dv'_{10}$$

$$v_{21} = Z_1((u_{21} - W_8)(u'_{11} - u_{21}) + D(u_{20} - u'_{10} - v'_{11}))$$

10. Finally, we adjust U_2 in step 10 of Table A.17.

$$u_{21} = u_{21}W_{11}Z_1; u_{20} = u_{20}W_{11}$$

6.2 Mixed Addition

While performing scalar multiplication on weighted coordinates, additions have to be performed after normalization of the precomputed point being added and the current result. If the precomputed points are also in a weighted form then the cost of normalization is normally at least twice that of the case in which the precomputed points were in affine form. Cohen, Miyaji and Ono [11] introduced the technique of reducing the precomputed points to affine form before the principal phase of the scalar multiplication. In order to utilize their technique, we derive formulae for mixed additions in which one point is represented in Lange's coordinates (or projective coordinates) and other point is represented in affine form.

6.2.1 Mixed Addition for Lange's New Coordinates

Given a divisor class $D_1 = (U_1, V_1, Z_{11}, z_{11} = Z_{11}^2, Z_{12}, z_{12}^2)$ in Lange's coordinates and another divisor class $D_2 = (U_2, V_2)$ in affine form, we obtain their sum D_3 represented in Lange's coordinates by the following steps. The formula is shown in Table A.12 of page 93.

1. We normalize of D_2 with respect to D_1 in steps 1-3 of Table A.12.

$$W_1 = Z_{11}Z_{12}; W_2 = z_{11}W_1;$$

$$u'_{20} = u_{20}z_{11}; u'_{21} = u_{21}z_{11}$$

$$v'_{20} = v_{20}W_2; v'_{21} = v_{21}W_2$$

2. We set up the linear system to find the monic form of S using the polynomial equation (Step 4 of Table A.12):

$$c(V'_2 - V_1) = (x + s'''_0)(U_1 - U'_2) - (u_{11} - u'_{21})U_2$$

The resulting linear system on the two variables (s_0''', c) is as follows.

$$a_{00} = u_{10} - u'_{20}; a_{01} = v_{10} - v'_{20}; q_0 = (u_{11} - u'_{21})u_{20};$$

$$a_{10} = u_{11} - u'_{21}; a_{11} = v_{11} - v'_{21}; q_1 = W_3 - a_{00};$$

where $W_3 = (u_{11} - u'_{21})u_{21}$.

We solve the linear system using Cramer's rule to obtain:

$$d = a_{00}a_{11} - a_{01}a_{10};$$

$$c' = dc = a_{00}q_1 - a_{10}q_0;$$

$$s_0'''' = ds_0''' = a_{11}q_0 - a_{01}q_1$$

Observe that from the fact that the difference in weights of U_1 and V_1 is $Z_{11}Z_{12}$:

$$s_0'''' = ds_0''$$

$$c' = dc = d/(Z_{11}Z_{12}s_1)$$

3. We perform some precomputations in step 5 of Table A.12.

$$D = d^2; W_4 = dc'; W_5 = c'W_1; W_6 = W_5Z_{11};$$

$$W_7 = W_5^2; W_8 = W_6^2; W_9 = s_0''''z_{11}; W_{10} = Z_{11}W_9;$$

4. We adjust V_1 in step 6 of Table A.12.

$$v'_{11} = W_4v_{11}; v'_{10} = W_4v_{10}$$

5. We use the expression for U_3 affine formula to compute U_3 with a weight of $(dZ_{11})^2$ (Step 7 of Table A.12).

$$u_{31} = 2W_{10} + Da_{10} - W_8$$

$$u_{30} = s_0''''(W_9 + 2a_{10}d) - Dq_1 + 2v'_{11} + (u'_{21} + u_{11})W_7$$

6. We adjust the weight of U_1. We obtain U'_1 with weight as that of U_3 (Step 8 of Table A.12).

$$u'_{11} = Du_{11}$$

$$u'_{10} = Du_{10}$$

7. We obtain the weights in step 9 of Table A.12.

$$Z_{31} = dZ_{11}; z_{31} = Z_{31}^2$$

$$Z_{32} = W_6; Z_{32} = W_8;$$

8. Using the normalized modular identity (Step 10 of Table A.12):

$$V_2 = -(z_{31}x + W_{10})U'_1 - z_{31}V'_1 \pmod{U}_3$$

we obtain V_3 as:

$$v_{30} = W_{10}(u_{30} - u'_{10}) + (u'_{11} - u'_{31})u_{30} - z_{31}v'_{10}$$

$$v_{31} = (u_{31} - W_{10})(u'_{11} - u_{21}) + z_{31}(u_{30} - u'_{10} - v'_{10})$$

6.2.2 Mixed Addition for Projective Coordinates

Given a divisor class $D_1 = (U_1, V_1, Z_1)$ in projective representation and another divisor class $D_2 = (U_2, V_2)$ in affine coordinates, we obtain their sum D_3 represented in projective coordinates by performing the following steps. The formula is presented in Table A.18 of page 99.

1. We normalize U_2 with respect to U_1 and V_2 with respect to V_1 (Step 1 of Table A.18).

$$u'_{20} = u_{20}Z_1; u'_{21} = u_{21}Z_1$$

$$v'_{20} = v_{20}Z_1; v'_{21} = v_{21}Z_1$$

2. Similar to the case of mixed addition for Lange's new coordinates, we set up the linear system to find monic(S) using the polynomial equation given below (Step 2 of Table A.18).

$$c(V'_2 - V_1) = (x + s'''_0)(U_1 - U'_2) - (u_{11} - u'_{21})U_2$$

The resulting linear system in variable (s'''_0, c') is as follows.

$$a_{00} = u_{10} - u'_{20}; a_{01} = v_{10} - v'_{20}; q_0 = (u_{11} - u'_{21})u_{20};$$

$$a_{10} = u_{11} - u'_{21}; a_{11} = v_{11} - v'_{21}; q_1 = W_0 - a_{00};$$

where $W_0 = (u_{11} - u'_{21})u_{21}$.
We solve the linear system using Cramer's rule to obtain the following values.

$$d = a_{00}a_{11} - a_{01}a_{10};$$

$$c' = dc = a_{00}q_1 - a_{10}q_0;$$

$$s''''_0 = ds'''_0 = a_{11}q_0 - a_{01}q_1$$

3. We perform some precomputation in step 3 of Table A.18.

$$D = d^2; W_1 = s''''_0 Z_1; W_2 = dW_1; W_3 = c'^2; W_4 = W_3 Z_1;$$

4. We compute V_1 with weight d^2Z_1/s_1 compared to its affine value in step 4 of Table A.18.

$$W_5 = dc'; v'_{11} = W_5 v_{11}; v'_{10} = W_5 v_{10}$$

5. We compute U_3 with weight DZ_1 (Step 5 of Table A.18).

$$u_{31} = 2W_2 + a_{10}D - W_4$$

$$u_{30} = s_0''''(W_1 + 2a_{10}d) + D(a_{00} - W_0) + 2v'_{11} + (u'_{21} + u_{11})W_3$$

6. We compute the final weight of the result in step 6 of Table A.18.

$$W_6 = DZ_1; W_7 = W_5 Z_1; Z_3 = W_6 W_7$$

7. We obtain U'_1 with weight DZ_1 in step 7 of Table A.18.

$$u'_{11} = Du_{11}$$

$$u'_{10} = Du_{10}$$

8. Using the normalized modular identity (Step 8 of Table A.18):

$$V_3 = -(W_6 x + W_2)U'_1 - W_6 V'_1 \pmod{U_3}$$

we obtain V_3 as follows:

$$v_{30} = W_2(u_{30} - u'_{10}) + (u'_{11} - u_{31})u_{30} - W_6 v'_{10}$$

$$v_{31} = (u_{31} - W_2)(u'_{11} - u_{31}) + W_6(u_{30} - u'_{10} - v'_{10})$$

9. We adjust the weight of U_3 in step 9 of Table A.18.

$$u_{31} = W_7 U_{31}; u_{30} = W_7 u_{30}$$

6.3 Double-Add Operation for Lange's New Coordinates

We obtain an efficient double-add operation compared to the naive method of performing doubling followed mixed addition for Lange's new coordinates. The technique we use was first proposed by Eisenträuger [13] for elliptic curves in affine coordinates. Eisenträuger, et al., obtain speed up by not explicitly computing polynomial V of intermediate divisor class. Fan and Gong adapted this technique to double-add operation on the Jacobian of genus two hyperelliptic curves when the coordinate system

used is affine coordinates. Longa and Miri [24], in the context of elliptic curves, optimized by observing that for the second addition normalization is not needed. We apply both techniques to genus two hyperelliptic curves for Lange's new coordinates. The formula is presented in Tables A.13 and A.14 of pages 94 and 95.

1. We perform the first eight steps of mixed addition. We reproduce those steps from mixed addition formula.

   (a) We normalize of D_2 with respect to D_1 in steps 1-3 of Table A.13.

   $$W_1 = Z_{11}Z_{12}; W_2 = z_{11}W_1;$$

   $$u'_{20} = u_{20}z_{11}; u'_{21} = u_{21}z_{11}$$

   $$v'_{20} = v_{20}W_2; v'_{21} = v_{21}W_2$$

   (b) We set up the linear system to find the monic form of S using the polynomial equation given below (Step 4 of Table A.13).

   $$c(V'_2 - V_1) = (x + s'''_0)(U_1 - U'_2) - (u_{11} - u'_{21})U_2$$

   The resulting linear system on two variables (s'''_0, c') as follows.

   $$a_{00} = u_{10} - u'_{20}; a_{01} = v_{10} - v'_{20}; q_0 = (u_{11} - u'_{21})u_{20};$$

   $$a_{10} = u_{11} - u'_{21}; a_{11} = v_{11} - v'_{21}; q_1 = W_3 - a_{00};$$

   where $W_3 = (u_{11} - u'_{21})u_{21}$.
   We solve the linear system using Cramer's rule to obtain:

   $$d = a_{00}a_{11} - a_{01}a_{10};$$

   $$c' = dc' = a_{00}q_1 - a_{10}q_0;$$

   $$s''''_0 = ds'''_0 = a_{11}q_0 - a_{01}q_1$$

   We observe from the fact that the difference in weights of U_1 and V_1 is

$Z_{11}Z_{12}$:

$$s_0'''' = ds_0''$$

$$c' = dc' = d/(Z_{11}Z_{12})$$

(c) We perform some precomputation in step 5 of Table A.13.

$$D = d^2; W_4 = dc'; W_5 = c'W_1; W_6 = W_5Z_{11};$$

$$W_7 = W_5^2; W_8 = W_6^2; W_9 = s_0''''z_{11}; W_{10} = Z_{11}W_9$$

(d) We adjust V_1 in step 6 of Table A.13.

$$v'_{11} = W_4v_{11}; v'_{10} = W_4v_{10}$$

(e) We use the expression for U_3 affine formula to compute U_3 with weight $(dZ_{11})^2$ (Step 7 of Table A.13).

$$u_{31} = 2W_{10} + Da_{10} - W_8$$

$$u_{30} = s_0''''(W_9 + 2a_{10}d) - Dq_1 + 2v'_{11} + (u'_{21} + u_{11})W_7$$

(f) We adjust the weight of U_1. We obtain U_1' with weight as that of U_3 in step 8 of Table A.13.

$$u'_{11} = Du_{11}$$

$$u'_{10} = Du_{10}$$

2. We compute weights Step 9 of Table A.13.

$$Z_{31} = dZ_{11}; z_{31} = Z_{31}^2$$

3. We set up the following linear system to compute $T = \frac{-2V_1'}{U_1'} \pmod{U_3}$ (Step 10

of Table A.14).

$$-2V_1' = \tilde{T}(U_1' - U_3) + \tilde{t}_1(u_{31} - u_{11}')U_3$$

where $\tilde{T} = \tilde{t}_1 z_{31} x + \tilde{t}_0$ By equating coefficients of powers of x we compute:

$$\tilde{a}_{00} = u_{10}' - u_{30}; \tilde{a}_{10} = u_{11}' - u_{31};$$

$$\tilde{q}_0 = -2v_{10}'; \tilde{q}_1 = -2v_{11}'$$

$$\tilde{a}_{10} = -\tilde{a}_{10}u_{30}; \tilde{a}_{11} = -\tilde{a}_{10}u_{31} + \tilde{a}_{00}z_{31}$$

By applying Cramer's rule, we compute:

$$\tilde{d}' = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{10}\tilde{a}_{01}$$

$$\tilde{t}_1'(= \tilde{d}'\tilde{t}_1) = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$$

$$\tilde{t}_0'(= \tilde{d}'\tilde{t}_0) = \tilde{a}_{11}\tilde{q}_0 - \tilde{a}_{01}\tilde{q}_1$$

Let $\tilde{T}' = \tilde{t}_1' z_{31} x + \tilde{t}_0'$.

4. Now we perform normalized addition of S'''' and $\tilde{T}'$ and some precomputation (Step 11 of Table A.14).

$$\tilde{s}_0'''' = \tilde{t}_0 z_{31} - W_{10}\tilde{d}';$$

$$\tilde{W}_0 = \tilde{d}' Z_{32}; \tilde{W}_1 = \tilde{W}_0 Z_{31}$$

$$\tilde{d}'' = \tilde{t}_1' z_{31} - \tilde{d}'$$

$$\tilde{d} = \tilde{d}'' z_{31}$$

$$\tilde{W}_2 = \tilde{d}'\tilde{d}$$

$$\tilde{W}_3 = \tilde{s}_0''''\tilde{d}$$

$$\tilde{W}_4 = \tilde{W}_1^2;$$

$$\tilde{W}_5 = \tilde{d}''\tilde{d}; \tilde{W}_6 = \tilde{d}''^2; \tilde{W}_7 = \tilde{W}_0^2;$$

5. We adjust V_1 in step 12 of Table A.14.

$$v'_{11} = v'_{11}\tilde{W}_2; \tilde{v}'_{10} = v'_{10}\tilde{W}_2$$

6. We compute U_4 with weight $\tilde{d}^2$ in step 13 of Table A.14.

$$u_{41} = 2\tilde{W}_3 - \tilde{W}_4 + \tilde{W}_5\tilde{a}_{10}$$

$$u_{40} = \tilde{s''''}_0(\tilde{s''''}_0 + 2\tilde{a}_{10}\tilde{d}'') + \tilde{a}_{11}\tilde{W}_6 + 2\tilde{v}'_{11} + (u_{31} + u'_{11})\tilde{W}_7$$

7. We obtain weights in step 14 of Table A.14.

$$Z_{41} = \tilde{d}; z_{41} = Z_{41}^2$$

$$Z_{42} = \tilde{W}_1, z_{42} = \tilde{W}_4$$

8. We compute a form of U_1 with the weight as that of U_4 (Step 15 of Table A.14).

$$\tilde{u}'_{11} = u'_{11}\tilde{W}_5, \tilde{u}'_{10} = u'_{10}\tilde{W}_5$$

9. Finally, we compute V_4 in step 16 of Table A.14.

$$v_{41} = (u_{41} - \tilde{W}_3)(\tilde{u}'_{11} - u_{41}) + z_{41}(u_{41} - \tilde{u}'_{11} - \tilde{v}'_{11})$$

$$v_{40} = \tilde{W}_3(u_{40} - \tilde{u}'_{10}) + (\tilde{u}'_{11} - u_{41})u_{40} - z_{41}\tilde{v}'_{10}$$

6.4 Tripling

We compute tripling by performing doubling followed by a addition for both Lange's new coordinates and projective coordinates. We side step computing the intermediate V polynomial. We apply the technique of computing the U polynomial of the result in the Mumford representation modulo U_1 where U_1 is the first polynomial of the

Mumford representation of the divisor to be tripled.

6.4.1 Lange's New Coordinates

We proceed as follows to triple a divisor given in Lange's new coordinates. The formula is presented in Tables A.15 and A.16 of pages 96 and 97.

1. We first perform the first 7 steps of doubling. For the sake of readability we reproduce those steps.

   (a) We compute $V_1' = 2V_1$ step 1 of Table A.15.

   (b) We compute $k_1 x + k_0 \equiv K \pmod{U_1}$ step 2 of Table A.15. We compute k_1 with weight $z_{11}^2 z_{12}$ and k_0 with weight $z_{11}^3 z_{12}$ as: .

$$w_0 = v_{11}^2$$

$$w_1 = u_{11}^2$$

$$W_2 = z_{11}^2$$

$$W_3 = u_{10} z_{11}$$

$$W_4 = f_3 W_2 + w_1$$

$$w_5 = 2W_3$$

$$k_1 = (2w_1 + w_4 - w_5) z_{12}$$

$$k_0 = (u_{11}(2w_5 - w_4) + f_2 W_2) z_{12} - w_0$$

   In further discussions we denote by k_1' the value $k_1 z_{11}$ which is not computed in the formula.

   (c) We set up the linear system for computation $K/2V_1 \pmod{U_1}$ using the polynomial equation given below. (Step 3 of Table A.15)

$$(k_1' x + k_0)c = (z_{11} x + s_0''')(2v_{11} x + 2v_{10}) - 2v_{11}(x^2 + u_{11} x + u_{10})$$

We note that $s_0''' = z_{11}s_0''$. By equating coefficients of identical powers of the indeterminate, we obtain the following linear system.

$$a_{00} = 2v_{10}; a_{01} = -k_0; q_0 = 2v_{11}u_{10};$$

$$a_{10} = 2v_{11}; a_{11} = -k_1'; q_1 = 2v_{11}u_{11} - 2v_0z_{11}$$

Instead of computing a_{00}, a_{11}, q_0 as above, we compute them as follows.

$$a_{00} = v_{10}'z_{11}; a_{11} = -k_1; q_0 = v_{11}'W_3$$

Now $q_1 = v_{11}'u_{11} - a_{00}$. The determinant of the linear system is computed as:

$$d = a_{00}a_{11} - a_{01}a_{10}$$

We obtain $c' = z_{11}dc$ and $s_0'''' = ds_0'''$ by computing

$$c' = a_{00}q_1 - 2v_{11}q_0$$

$$s_0'''' = q_0a_{11} - q_1a_{01}$$

If $c' = 0$ then $\gcd(U_1, V_1)$ is not zero. If the determinant d is zero then s_1 is zero. These cases occur rarely. These cases can be handled by Cantor's algorithm without significant performance penalty. We do not consider these cases. We observe that :

$$c' = dz_{11}c' = (dz_{11}^2)(1/(Z_{11}^3Z_{12}))/s_1 = (dZ_{11})/(Z_{12}s_1)$$

It follows from this that there is a weight of $1/(Z_{11}^3Z_{12})$ introduced by the differences in weights of the denominator and numerator in the computation of S and a weight of z_{11} is introduced by the fact that z_{11} is the leading coefficient of the modular fraction in the linear system. Further s_0'''' and its affine value s_0'' are related are shown next.

$$s_0'''' = ds_0''' = dz_{11}s_0''$$

(d) We perform some precomputation on step 4 of Table A.15.

$$W_6 = c'Z_{12}; W_7 = W_6 Z_{11};$$

$$W_8 = dz_{11}; W_9 = W_8^2$$

$$W_{10} = W_7^2; W_{11} = W_8 s_0'''$$

We note that $W_6 = (dZ_{11})/s_1$ and $W_7 = (dz_{11})/s_1$.

(e) We compute $V_1' = W_9 V_1/s_1$ as (Step 5 of Table A.15):

$$W_{12} = dc'; v_{11}' = W_{12} v_{11}; v_{10}' = W_{12} v_{10}$$

$dc' = (d^2 Z_{11})/(Z_{12} s_1)$. Since we have a weight of $Z_{11}^3 Z_{12}$ on V we obtain V' with required weight.

(f) We obtain U_2 with weight W_9 in step 6 of Table A.15.

$$u_{21} = 2W_{11} - W_{10}$$

$$u_{20} = s_0'''^2 + 2v_{11}' + 2u_{11} W_6^2$$

(g) We adjust U_1. We compute a form of U_1 with weight as that of U_2 (Step 7 of Table A.15).

$$W_{13} = W_8 d; u_{11}' = W_{24} u_{11}; u_{10}' = W_{13} u_{10}$$

2. We compute weights in step 8 of Table A.15.

$$Z_{21} = W_8, z_{21} = W_9$$

$$Z_{22} = W_7, z_{22} = W_{10}$$

3. We compute a weighted $\tilde{T} = -2V_1'/U_2 \pmod{U_1}$ in step 9 of Table A.16. We

set up the linear system as

$$-2V_1' = \tilde{T}(U_2 - U_1) + (u_{11} - u_{21})U_1$$

where $\tilde{T} = z_{21}\tilde{t}_1 x + \tilde{t}_0$. We solve the system up to a factor of the determinant using Cramer's rule.

$$\tilde{a}_{00} = u_{20} - u_{10}'; \tilde{a}_{10} = u_{21} - u_{11}'$$

$$\tilde{a}_{01} = -2v_{10}'; \tilde{a}_{11} = -2v_{11}$$

$$\tilde{w}_0 = \tilde{a}_{00}z_{21}, \tilde{q}_0 = \tilde{a}_{10}u_{10}', \tilde{q}_{11} = \tilde{a}_{10}u_{11}' - \tilde{w}_0$$

$$\tilde{W}_1 = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}, \tilde{t}_1' = z_{11}\tilde{W}_1$$

$$\tilde{t}_0' = \tilde{a}_{11}\tilde{q}_0 - \tilde{a}_{01}\tilde{q}_1$$

$$\tilde{d}' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$$

4. We compute a weighted $\tilde{R} = 2V_1'/U_1 \pmod{U_2}$ in step 10 of Table A.16. We observe that:

$$-2V_1 = \tilde{T}U_2 - \tilde{R}U_1$$

Using the above relation we compute

$$\tilde{r}_1' = \tilde{t}_1', \tilde{r}_0' = \tilde{t}_0' + \tilde{W}_1(u_{21} - u_{11}')$$

5. We perform some precomputation and compute a weighted $\tilde{S} = (V_2 - V_1)/U_1 \pmod{U_2}$ by performing normalized addition of $\tilde{T}$ and S'' (Step 11 of Table A.16).

$$\tilde{W}_3 = \tilde{r}'; \tilde{W}_4 = z_{21}\tilde{r}_0'$$

$$\tilde{W}_5 = W_{11}\tilde{d}'; \tilde{s}_0' = \tilde{W}_4 - \tilde{W}_5;$$

6. We compute a weighted $\bar{S} = S'' - \tilde{S}$ by normalized addition (Step 12 of Table

A.16).

$$\bar{s}'_1 = \tilde{r}' - 2\tilde{d}', \bar{s}_0 = -2\tilde{W}_5 + \tilde{W}_4$$

At this point, we observe that $\bar{s}_1 = z_{21}\bar{s}'_1$.

7. We compute $U'_3 = U_3 \pmod{U_1}$ using Equation 4.6 (Step 13 of Table A.16)

$$\tilde{W}_6 = \tilde{t}'_1\bar{s}_1, u'_{30} = \bar{s}_0\tilde{t}_0 - \tilde{W}_6 u'_{10}$$

Subsequently, we compute u_{31} exactly using the expression given in Table 2.2 in order to save some computation.

8. We compute U_3 (Step 14-16,18 of Table A.16). By noting that the weight of U'_3 is $\tilde{W}_3^2 z_{21}$, we compute U_3 and the weights as follows:

$$\tilde{W}_7 = \tilde{W}_3^2$$

$$u_{30} = u'_{30} + \tilde{W}_7 u'_{10}$$

$$Z_{31} = \tilde{W}_3 Z_{21}, z_{31} = Z_{31}^2$$

$$Z_{32}(= Z_{31}/s_1) = \tilde{d}' Z_{22}$$

$$\tilde{W}_8 = \tilde{W}_3\tilde{s}'_0; \tilde{W}_9 = \tilde{W}_3\tilde{d}'$$

$$u_{31} = 2\tilde{W}_8 + (u'_{11} - u_{21})\tilde{W}_7 - z_{32}; u'_{31} = u_{31} - u'_{11}\tilde{W}_7;$$

9. We adjust V_1 in step 17 of Table A.16.

$$\tilde{v}'_{11} = \tilde{W}_9 v'_{11}, \tilde{v}'_{10} = \tilde{W}_9 v'_{10}$$

10. Finally, we obtain V_3 as step 19 of Table A.16.

$$v_{31} = -(u_{31} - \tilde{W}_8)u'_{31} + z_{31}(u'_{30} - \tilde{v}_{11})$$

$$v_{30} = \tilde{W}_8 u'_{30} - u'_{31}u_{30} - z_{31}\tilde{v}'_{10}$$

6.4.2 Projective Coordinates

To perform tripling of reduced divisor given in projective coordinates we proceed as shown subsequently. The formula is presented in Tables A.19 and A.20 of pages 100 and 101.

1. We first perform the first eight steps of doubling with modifications.

   (a) We compute $V_1' = 2V_1$ in step 1 of Table A.19.

   (b) Similar to the case of Lange's coordinates we compute k_1 and k_0 with different weights. We compute k_1 with the weight Z_1^2 and k_0 with the weight Z_1^3 as presented below (Step 2 of Table A.19).

$$w_0 = v_{11}^2$$

$$w_1 = u_{11}^2$$

$$W_2 = Z^2$$

$$w_3 = f_2 W_2$$

$$W_4 = u_{10} Z_1$$

$$w_5 = f_3 W_2 + w_1$$

$$w_6 = 2W_4$$

$$k_1 = 2w_1 + w_3 - w_5$$

$$k_0 = u_{11}(2w_5 - w_3) + (w_3 - w_0)Z_1$$

   In further discussion we denote by k_1' the value $k_1 Z_1$ which was not computed in the formula.

   (c) A linear system for computation of $K/2V_1 \pmod{U_1}$ is set up using the

polynomial equation shown in what follows (Step 3 of Table A.19).

$$(k_1'x + k_0)c = (Zx + s_0''')(2v_{11}x + 2v_{10}) - 2v_{11}(x^2 + u_{11}x + u_{10})$$

By equating the coefficients of powers of x, we obtain a system of two linear equations.

$$a_{00} = v_{10}'; a_{01} = -k_0; q_0 = v_{11}'u_{10};$$

$$a_{10} = v_{11}'; a_{11} = -k_1'; q_1 = v_{11}'u_{11} - v_0'Z_1$$

Instead of computing a_{00}, a_{11}, q_0 as above, we perform the following steps.

$$a_{00} = v_{10}'Z_1; a_{11} = -k_1; q_0 = v_{11}'W_4; a_{01} = -k_0; a_{10} = v_{11}'$$

We derive q_1 as $w_7 - a_{00}'$ where $w_7 = v_{11}'u_{11}$. The determinant of the linear system is computed as:

$$d = a_{00}a_{11} - a_{01}a_{10}$$

We obtain c' and s_0'''' with factors of dZ and d compared respectively to c and s_0''' by computing

$$c' = a_{00}q_1 - a_{10}q_0$$

$$s_0'''' = q_0a_{11} - q_1a_{01}$$

If $c' = 0$ then $\gcd(U_1, V_1)$ is not zero. As previously mentioned, the infrequent cases of $d = 0$ and/or $c' = 0$ are not considered in this discussion.

(d) We precompute the following values (Step 4 of Table A.19).

$$Z_{21} = dZ_1; z_{21} = Z_{21}^2;$$

$$W_8 = Z_{21}s_0''''$$

$$w_9 = c'Z_1$$

$$W_{10} = c'w_9$$

$$W_{11} = dw_9$$

$$Z_{22} = w_9$$

$$z_{22} = Z_{22}^2$$

(e) We adjust V_1 in step 5 of Table A.19.

$$v'_{11} = W_{11}v_{11}$$

$$v'_{10} = W_{11}v_{10}$$

V'_1 has a weight $d^2Z_1^2$ w.r.t the corresponding affine value.

(f) We compute U_2 in step 6 of Table A.19.

$$u_{21} = 2W_8 - z_{22}$$

$$u_{20} = 2u_{11}W_{10} + 2v'_{11} + s_0''''^2$$

In this step we compute U_2 with a weight of DZ_1^2.

(g) We adjust U_1 in Step 7 of A.19. In this step we compute u_{10} and u_{11} with weights as that of u_{20} and u_{21} respectively.

$$w_{12} = dZ_{21}; u'_{11} = w_{12}u_{11}; u'_{10} = w_{12}u_{10}$$

2. At this stage, we perform an addition to obtain the result in Lange's new coordinates

(a) We compute $\tilde{T} = -2V'_1/U_2 \pmod{U_1}$ in step 8 of Table A.20. We set up the linear system as

$$-2V'_1 = \tilde{T}(U_2 - U_1) + (u_{11} - u_{21})U_1$$

where $\tilde{T} = z_{21}\tilde{t}_1 x + \tilde{t}_0$. We solve the system up to a factor of the determinant using Cramer's rule.

$$\tilde{a}_{00} = u_{20} - u'_{10}; \tilde{a}_{10} = u_{21} - u'_{11}$$

$$\tilde{a}_{01} = -2v'_{10}; \tilde{a}_{11} = -2v_{11}$$

$$\tilde{w}_0 = \tilde{a}_{00}z_{21}, \tilde{q}_0 = \tilde{a}_{10}u'_{10}, \tilde{q}_{11} = \tilde{a}_{10}u'_{11} - \tilde{w}_0$$

$$\tilde{W}_1 = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}, \tilde{t}'_1 = z_{11}\tilde{W}_1$$

$$\tilde{t}'_0 = \tilde{a}_{11}\tilde{q}_0 - \tilde{a}_{01}\tilde{q}_1$$

$$\tilde{d}' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$$

(b) We Compute the weighted $\tilde{R} = 2V'_1/U_1 \pmod{U_2}$ (Step 9 of Table A.20). We observe that:

$$-2V_1 = \tilde{T}U_2 - \tilde{R}U_1$$

Using the above relation we compute as

$$\tilde{r}'_1 = \tilde{t}'_1, \tilde{r}'_0 = \tilde{t}'_0 + \tilde{W}_1(u_{21} - u'_{11})$$

(c) We perform some precomputation and compute a weighted $\tilde{S} = (V_2 - V_1)/U_1 \pmod{U_2}$ by performing normalized addition of $\tilde{T}$ and S'' (Step 10 of Table A.20).

$$\tilde{W}_3 = \tilde{r}'; \tilde{W}_4 = z_{21}\tilde{r}'_0$$

$$\tilde{W}_5 = W_{11}\tilde{d}'; \tilde{s}'_0 = \tilde{W}_4 - \tilde{W}_5;$$

(d) We compute a weighted $\bar{S} = S'' - \tilde{S}$ by normalized addition (Step 11 of Table A.20).

$$\bar{s}'_1 = \tilde{r}' - 2\tilde{d}', \bar{s}_0 = -2\tilde{W}_5 + \tilde{W}_4$$

At this point, we observe that $\bar{s}_1 = z_{21}\bar{s}'_1$.

(e) We compute $U'_3 = U_3 \pmod{U_1}$ using the Equation 4.6 (Step 12 of Table

A.20)

$$\tilde{W}_6 = \tilde{t}'_1 \tilde{s}_1, u'_{30} = \bar{s}_0 \tilde{t}_0 - \tilde{W}_6 u'_{10}$$

We compute u_{31} exactly using the expression given in the Table 2.2 in order to save some computation.

(f) We obtain U_3 in steps 13-15,17 of Table A.20. By noting that the weight of U'_3 is $\tilde{W}_3^2 z_{21}$, we compute U_3 and weights as follows:

$$\tilde{W}_7 = \tilde{W}_3^2$$

$$u_{30} = u'_{30} + \tilde{W}_7 u'_{10}$$

$$Z_{31} = \tilde{W}_3 Z_{21}, z_{31} = Z_{31}^2$$

$$Z_{32} (= Z_{31}/s_1) = \tilde{d}' Z_{22}$$

$$\tilde{W}_8 = \tilde{W}_3 \tilde{s}'_0; \tilde{W}_9 = \tilde{W}_3 \tilde{d}'$$

$$u_{31} = 2\tilde{W}_8 + (u'_{11} - u_{21})\tilde{W}_7 - z_{32}; u'_{31} = u_{31} - u'_{11}\tilde{W}_7;$$

(g) We adjust V_1 in Step 16 of Table A.20.

$$\tilde{v}'_{11} = \tilde{W}_9 v'_{11}, \tilde{v}'_{10} = \tilde{W}_9 v'_{10}$$

(h) We compute V_3 in step 18 of Table A.20.

$$v_{31} = -(u_{31} - \tilde{W}_8)u'_{31} + z_{31}(u'_{30} - \tilde{v}_{11})$$

$$v_{30} = \tilde{W}_8 u'_{30} - u'_{31} u_{30} - z_{31}\tilde{v}'_{10}$$

3. Finally, we adjust the weight of U_3 and compute Z_3 (Step 19 of Table A.20).

$$\tilde{w}_{10} = Z_{31} Z_{32}; Z_3 = z_{31}\tilde{w}_{10};$$

$$u_{31} = u_{31}\tilde{W}_{10}; u_{30} = u_{30}\tilde{w}_{10}$$

6.5 Precomputation Schemes and Comparison of Performance Against Existing Methods

In this section, we compare the cost of scalar multiplication with our new and improved formulae to that with scalar multiplication using existing formulae. We investigate the improvements using the nuMongo library. In that nuMongo implementation the cost of an inversion is equal to the cost of 29.5 multiplications and that of squaring operation is equal to one multiplication.

Special Addition and Precomputation

First we develop a special addition of two reduced divisors in projective coordinates with identical weights. This technique was first used by Longa and Miri for elliptic curves [23].

Given two reduced divisors with identical weights the formula two divisors $D_1 = (U_1, V_1, Z)$ and $D_2 = (U_2, V_2, Z)$ the formula computes $D_3 = D_1 + D_2 = (U_3, V_3, Z_3)$. A form, in projective coordinates, of D_1, (U_1', V_1', Z_3) with weight Z_3 is computed. $\frac{Z_3}{Z}$ is also computed by formula. The formula is presented in Table A.21 of page 102.

1. We set up the linear system and solve it in step 2 of Table A.21.

$$a_{00} = u_{10} - u_{20}, a_{10} = u_{11} - u_{21}$$

$$a_{01} = v_{10} - v_{20}, a_{11} = v_{11} - v_{21}$$

$$q_0 = u_{20}a_{10}, W_0 = a_{10}u_{21}, q_0 = W_0 - a_{00}Z$$

$$d' = a_{00}a_{11} - a_{01}a_{10}, d = dZ$$

$$s_0''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - a_{10}q_0$$

2. We perform some precomputation in step 3 of Table A.21.

$$D = d^2, W_1 = s_0'''Z, W_2 = dW_1, W_3 = c'^2, W_4 = W_3Z$$

3. We adjust V_1 in step 4 of Table A.21. V_1' is computed as the affine value of

V_1/s_1 multiplied by DZ_1.

$$W_5 = dc_1', v_{11}' = W_5 v_{11}, v_{10}' = W_5 v_{10}$$

4. We compute U_3 with weight DZ_1 in step 5 of Table A.21.

$$u_{31} = 2W_2 + a_{10}D - W_4$$

$$u_{30} = s_0'''(W_1 + 2da_{10}) - dd'q_1 + 2v_{11}' + (u_{21}' + u_{11})W_3$$

5. We compute Z_3, Z_3' in step 6 of Table A.21.

$$W_6 = DZ_1, W_7 = W_5 Z_1, Z_3 = W_6 W_7, Z_3' = W_5 W_6$$

6. We adjust U_1 so as to be the same as U_3 in step 7 of Table A.21.

$$u_{11}' = Du_{11}, u_{10}' = Du_{10}$$

7. We adjust V_1 to have a weight of Z_3 in step 8 of Table A.21.

$$v_{11}' = v_{11}' W_6, v_{10}' = v_{11}' W_6$$

8. We compute V_3 with weight Z_3 in Step 9 of Table A.21.

$$v_{30} = W_2(u_{30} - u_{10}') + (u_{11}' - u_{31})u_{30} - v_{10}'$$

$$v_{31} = (u_{31} - W_2)(u_{11}' - u_{31}) + W_6(u_{30} - u_{10}') - v_{11}'$$

9. We adjust U_3, U_1 so that they have weight Z_3 in step 10 of Table A.21.

$$u_{31} = W_7 u_{31}, u_{30} = W_7 u_{30}, u_{11}' = u_{11}' W_7, u_{10}' = u_{10}' W_7$$

Precomputation for w-NAF scalar multiplication is performed as follows:

1. Let D be the base point to be multiplied by scalar.
2. Compute $2D$ (Doubling operation cost).
3. Compute D with the weight as that of $2D$ (3M).
4. Perform special additions to compute $3D, 5D, \cdots, kD$(one special addition per point).
5. Invert Z coordinate of kD(1M).
6. Obtain inverse of Z coordinates of $k-2, k-4, \cdots, 3$(1M per point).
7. Remove weights on $3D, 5D, \cdots, kD$ (4M per point).

The total cost is equal to Cost of Doubling of Point in Affine Coordinates+ 3 M+$(2^{w-2}-1)\times$ (Cost of Special Addition)+I+ (5M) $\times(2^{w-2}-1)$. Table 6.1 compares the cost of precomputation by our method with that of the existing method shown in [22].

6.5.1 Lange's New Coordinates

In this section, we compare the cost of scalar multiplication using our formulae against existing formulae for Lange's new coordinates. Precomputation is performed on projective coordinates. The comparison of cost of fomulae is presented in Table 6.2. We compare the cost of our traditional formulae against the results presented in [22]. Composite formulae are compared against the cost of naive methods of computing composite operations using doubling and addition.

Table 6.3 shows the speed up obtained for the main phase of scalar multiplication and the total cost (including precomputation).

Table 6.4 shows the total cost of scalar multiplication for a 256-bit scalar using binary expansions of scalars. Table 6.5 shows the same for the multibase method. But we compare the cost using multibase methods against our binary methods. We use only (2,3)-wmbNAF methods with zero window on base 3. This is because tripling is not efficient enough compared to the double-add operation. This is in contrast to the case of genus one curves for which multibase methods are used to obtain significant speed up. We observe the following principal features of improvements:

Table 6.1: Costs of Precomputations on Projective Coordinates

Window	This Work	Previous
2	-	-
3	I+61M+9SS	I+89M+6S
4	I+164M+13S	I+197M+10S

Table 6.2: Costs of Formulae for Lange's New Coordinates

Formulae	This Work	Previous Work
Doubling	30M+7S	34M+7S
Mixed Addition	34M+4S	37M+6S
Double-Add	61M+8S	71M+13S
Tripling	55M+10S	71M+13S

1. We obtain up to 13.85% speed up for the main phase using binary expansions of scalars.

2. Up to 13.11% improvement in the total cost using binary expansions of scalars.

3. Total cost is improved by up to 2.9% using multibase methods compared with that of our binary expansion methods

4. Overall improvement due to multibase methods is up to 16% compared with existing methods.

6.5.2 Projective Coordinates

Comparison costs of formulae are presented in Table 6.6. We compare traditional formulae against results presented in [22]. Composite formulae are compared against the naive method cost of computing composite operations using doubling and addition. Table 6.7 compares the cost of the main phase of scalar multiplication using

Table 6.3: Costs of Main Phase per Bit Scalar using binary methods for Lange's New coordinates

Window	Existing	This work	Reduction
2	(139M+27S)/3	(121M+22S)/3	13.85%
3	(173M+34S)/4	(151M+29S)/4	13.04%
4	(207M+41S)/5	(181+36S)/5	12.50%

Table 6.4: Costs for Scalar Multiplications on Lange's New Coordinates Using Binary Methods for 256-Bit Scalar

Window	**Existing**	**This work**	**Reduction**
3	13373M	11620M	13.11%
4	12934M	11317M	12.5%

Table 6.5: Costs for Scalar Multiplications on Lange's New Coordinates Using (2,3)-$wmbNAF_w$ methods for 256-Bit Scalar

Weight(w)	**Binary (This Work)**	**This work($wmbNAF$)**	**Reduction**
2	12203M	11850M	2.9%
3	11620M	11456M	1.4%

our methods for Lange's new coordinates and projective coordinates. Scalar multiplication on projective coordinates is slower by less than 3.3% compared with scalar multiplication on Lange's new coordinates. We conclude that projective coordinates are as effective as scalar multiplication on Lange's new coordinates. We conclude that projective coordinates are almost as efficient as Lange's new coordinates.

Table 6.6: Costs of Formulae for Projective Coordinates

Formulae	**This Work**	**Existing**
Doubling	32M+5S	38M+7S
Mixed Addition	35M+2S	40M+4S
Tripling	57M+9S	67M+7S

Table 6.7: Costs per Bit Scalar for Projective Coordinates

Window	**Projective coordinates**	**Lange's New Coords.**	**Slower By**
2	(131M+17S)/3	(121M+22S)/3	3.3%
3	(163M+22S)/4	(151M+29S)/4	2.7%
4	(195M+27S)/5	(181M+36S)/5	2.3%

Chapter 7

Conclusions and Future Work

Scalar multiplication is the most important operation in hyperelliptic cryptosystems. In this work, we mainly improved scalar multiplication by decreasing the computational time. This work focused primarily on the point arithmetic level. Improvements were made on coordinates which use field inversion as well as for inversion-free coordinates. Contribution was made on space requirements by obtaining formulae in projective coordinates which need a single weight value per point; these are almost as efficient as formulae on Lange's new coordinates. Previously formulae in Lange's new coordinate were significantly faster than those of projective coordinates. We introduced semi-affine coordinates. Formulae for semi-affine coordinates require field inversions too but are significantly faster than that of existing affine coordinates. We applied multibase representations with the tripling formulae to further speed up scalar multiplication. We also improved precomputation cost by using efficient double-add operations and special addition.

One future direction would be the application of techniques used in this thesis to hyperelliptic curves of higher genera. Genus 3 hyperelliptic curves over prime fields with 64-bit modulus provides the security of elliptic curves over 192-bit field. With the widespread use of 64-bit architectures, significant improvement in point formulae in genera 3 and 4 in conjunction with the use of machine instructions to perform field operation may lead to more efficient cryptosystems. We note that when we multiply two 64-bit numbers the number of bits in the result can be upto 128 bits. But applying technique of Karatsuba multiplication and viewing a 64-bit integer as a polynomial with coefficients of size 32-bit we can implement multiplication followed by modular reduction using 64-bit machine instructions.

In order to construct practical cryptosystem based on hyperelliptic curves, atomic

formulae which resist the attacks based on side channel information have to be developed. We have in this thesis specialized computation by using double-add and triple formulae. As we specialize operations to save computation adversary will have access to more side-channel information. Hence protection against side-channel attacks become important. For elliptic curves with fewer operations it is not hard to obtain efficient formulae in the form of atomic blocks which will have similar execution trace. But in the case of hyperelliptic curves with a significantly larger number of field operations, efficient atomic formulae would only be possible using some kind of automation of search for atomic blocks. The problem can be converted to coloured graph traversal problems as follows:

1. We build a directed graph with dependency information for each formula.

2. Nodes are initial, final or intermediate variables will be coloured based on the operation used to obtain them.

3. A directed edge (v, w) is in the edge set of graph iff v is used to obtain w. There may be more than one possible graph for an operation.

4. A formulae in the form series of atomic blocks, will have a pre-order traversal of a graph as subsequence of sequence of operations. Hence search space can be bounded.

In genus two hyperelliptic curves over a prime field, computation $K \pmod U$ requires many operations for normalization. Our initial inspection shows that using expression $K = ((f(X) - (-SU' - V_1 + S'U)/U)$ when the previous operation is doubling, $D = 2D'$ and S denotes the usual polynomial calculated for doubling, will give some gain on Lange's new coordinates.

In this thesis we demonstrated that computation can be saved by identifying relationships between values in adjacent traditional operations. We observe that for genus two hyperelliptic curves over binary fields:

1. There is no division by V in any of the doubling formulae. Further there is no multiplication by V for doubling in the binary case.

2. Squaring distributes over the polynomial addition for polynomials over binary fields.

3. Some of the coefficients of V can be recovered by the fact $U|(f(X) + h(X)V + V^2)$.

Based on these observations computation across point operations can be saved by avoiding computing some coefficients of V until addition. When addition needs to be performed those coefficients can be recovered by the relation $U|(f(X)+h(X)V+V^2)$.

Appendix A

Table A.1: Doubling for Semi-Affine Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12},$z_{12} = Z_{12}^2$, $(U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}X + \bar{v}_{10}$	
	Output:$(U_2, \bar{V}_2 = Z_{22}V_2, Z_{22}, z_{22})$ where $D_2 = (U_2, V_2) = 2D_1$ is a reduced divisor $U_2 = X^2 + u_{21}X + u_{20}$,$\bar{V}_2 = \bar{v}_{21}X + \bar{v}_{20}$	
1	Compute $\bar{V}' = 2\bar{V}_1 \pmod{U_1}$ $v'_{11} = 2\bar{v}_{11}, v'_{10} = 2\bar{v}_{10}$	-
2	Precomputations $W_0 = \bar{v}_{11}^2, W_1 = u_{11}^2$	2S
3	Compute $K = (z_{12}f - \bar{V}^2)/U_1 \pmod{U_1} = k_1X + k_0$ $w_3 = f_3 + W_1, w_4 = 2u_{10}, k_1 = (2W_1 + w_3 - w_4)z_{12}$ $k_0 = (u_{11}(2w_4 - w_3) + f_2)z_{12} - W_0$	3M
4	Compute coefficients and determinant d the matrix : $a_{00} = v'_{10}, a_{01} = -k_0, , a_{10} = v'_{11}, a_{11} = -k_1$ $d = a_{00}a_{11} - a_{01}a_{10}$ $q_0 = v'_{11}u_{10}, q_1 = v'_{11}u_{21} - v'_{10}$ if $d = 0$ then different case	4M
5	Compute $S'' = X + s''_0 \equiv K/(2V_1s_1) \pmod{U_1})$ and Z_{22}: $i = d^{-1}, c(= 1/Z_{11}s_1) = (a_{00}q_1 - a_{10}q_0)i, Z_{22} = Z_{12}c$ $s''_0 = (a_{11}q_0 - a_{01}q_1)i, z_{22} = Z_{22}^2$ if $c = 0$ then different case	I+7M+1S
6	Precomputations $\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$ $\bar{v}_{11} = \bar{v}_{11}c = v_{11}Z_{22}, \bar{v}_{10} = \bar{v}_{10}c = v_{10}Z_{22}$	2M
7	Compute $U_2 = (S(SU_1 + 2V_1) - K)/U_1 = X^2 + u_{21}X + u_{20}$ $u_{20} = s''^2_0 + 2\bar{v}_{11} + 2u_1z_{22}$ $u_{21} = 2s''_0 - z_{22}$	1M+1S
8	Compute $\bar{V}_2 = Z_{22}V_2$ $\bar{v}_{21} = (u_{21} - s''_0)(u_{11} - u_{21}) + (u_{20} - u_{10}) - \bar{v}_{11}$ $\bar{v}_{20} = s''_0(u_{20} - u_{10}) + (u_{11} - u_{21})u_{20} - \bar{v}_{10}$	3M
	Total Cost I+20M+4S	

Table A.2: Mixed Addition for Semi-Affine Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12}, $z_{12} = Z_{12}^2, (U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor reduced divisor $D_2 = (U_2, V_2)$ $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}X + \bar{v}_{10}$ $U_2 = X^2 + u_{21}X + u_{20}$,$V_1 = v_{21}X + v_{20}$	
	Output: $(U_3, \bar{V}_3 = Z_{32}V_3)$, Z_{32} and $z_{32} = Z_{32}^2$ where $D_3 = (U_3, V_3) = D_1 + D_2$ is a reduced Divisor $U_3 = X^2 + u_{31}X + u_{30}, \bar{V}_3 = \bar{v}_{31}X + \bar{v}_{30}$	
1	Compute $\bar{V}_2 = Z_{12}V_2$ $\bar{v}_{21} = Z_{12}v_{21}$,$\bar{v}_{20} = Z_{12}v_{20}$	2M
2	Compute $S'' = (\bar{V}_2 - \bar{V}_1)/(s_1U_1) \pmod{U_2}$ and Z_{32} $a_{00} = u_{10} - u_{20}, a_{01} = \bar{v}_{10} - \bar{v}_{20}$ $a_{10} = u_{11} - u_{21}, a_{11} = \bar{v}_{11} - \bar{v}_{21}$ $d = a_{00}a_{11} - a_{01}a_{10}$,if $d = 0$ then different case,$i = d^{-1}$ $W_0 = u_{21}a_{10}, q_0 = u_{20}a_{10}, q_1 = W_0 - a_{00}$ $s_0'' = (a_{11}q_0 - a_{01}q_1)i, c = 1/(Z_{12}s_1) = (a_{00}q_1 - a_{10}q_0)i$ $Z_{32} = cZ_{12}, z_{32} = Z_{32}^2$ if $c = 0$ then different case	I+11M+1S
3	Precomputations $\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$ $\bar{v}_{11} = \bar{v}_{11}c = v_{11}Z_{32}, \bar{v}_{10} = \bar{v}_{10}c = v_{10}Z_{32}$	2M
4	Compute $U_3 = (S(L + 2V_1) - K)/U_2 = X^2 + u_1'X + u_0'$: $u_{30} = s_0''(s_0'' + 2a_{10}) - q_1 + 2\bar{v}_{11} + (u_{11} + u_{21})z_{32}$ $u_{31} = 2s_0'' + a_{10} - z_{32}$	2M
5	Compute $\bar{V}_3 = Z_{32}V_3$ $\bar{v}_{31} = (u_{31} - s_0'')(u_{11} - u_{31}) + (u_{30} - u_{10}) - \bar{v}_{11}$ $\bar{v}_{30} = s_0''(u_{30} - u_{10}) + (u_{11} - u_{31})u_{30} - \bar{v}_{10}$	3M
	Total Cost I+20M+1S	

Table A.3: Double-Add with Two Inversions for Semi-Affine Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12}, $z_{12} = Z_{12}^2, (U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor reduced divisor $D_2 = (U_2, V_2)$ $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}X + \bar{v}_{10}$ $U_2 = X^2 + u_{21}X + u_{20}$,$V_1 = v_{21}x + v_{20}$	
	Output: $(U_4, \bar{V}_4 = Z_{42}V_4)$, Z_{42} and $z_{42} = Z_{42}^2$ where $D_4 = (U_4, V_4) = 2D_1 + D_2$ is a reduced Divisor $U_4 = X^2 + u_{41}X + u_{40}, \bar{V}_4 = \bar{v}_{41}X + \bar{v}_{40}$	
1	Compute $\bar{V}_2 = Z_{12}V_2$ $\bar{v}_{21} = Z_{12}v_{21}$,$\bar{v}_{20} = Z_{12}v_{20}$	2M
2	Compute $S'' = (\bar{V}_2 - \bar{V}_1)/(s_1U_1) \pmod{U_2}$ and Z_{32} $a_{00} = u_{10} - u_{20}, a_{01} = \bar{v}_{10} - \bar{v}_{20}$ $a_{10} = u_{11} - u_{21}, a_{11} = \bar{v}_{11} - \bar{v}_{21}$ $d = a_{00}a_{11} - a_{01}a_{10}$,if $d = 0$ then different case,$i = d^{-1}$ $W_0 = u_{21}a_{10}, q_0 = u_{20}a_{10}, q_1 = W_0 - a_{00}$ $s_0'' = (a_{11}q_0 - a_{01}q_1)i, c = 1/(Z_{12}s_1) = (a_{00}q_1 - a_{10}q_0)i$ if $c = 0$ then different case $Z_{32} = cZ_{12}, z_{32} = Z_{32}^2$	I+11M+1S
3	Precomputations $\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$ $\bar{v}_{11} = \bar{v}_{11}c(= v_{11}Z_{32}), \bar{v}_{10} = \bar{v}_{10}c(= v_{10}Z_{32})$	2M
4	Compute $U_3 = (S(L + 2V_1) - K)/U_2 = X^2 + u_1'X + u_0'$: $u_{30} = s_0''(s_0'' + 2a_{10}) - q_1 + 2\bar{v}_{11} + (u_{11} + u_{21})z_{32}$ $u_{31} = 2s_0'' + a_{10} - z_{32}$	2M

Table A.4: Double-Add with Two Inversions for Semi-Affine Coordinates-Part 2

5	Compute $\tilde{S}'' = (\bar{V}_3 - \bar{V}_1)/(\tilde{s}_1 U_1) \pmod{U_3}$ and $Z_{32} = 1/s_1$ $\tilde{a}_{00} = u_{10} - u_{30}, \tilde{a}_{01} = 2\bar{v}_{10}, \tilde{q}_0 = u_{30}(u_{11} - u_{31})$ $\tilde{a}_{10} = u_{11} - u_{31}, \tilde{a}_{11} = 2\bar{v}_{11}, \tilde{W}_0 = u_{31}(u_{11} - u_{31})$ $\tilde{d}' = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}, \tilde{q}_1 = \tilde{W}_0 - u_{10} + u_{30}$ if $\tilde{d}' = 0$ then different case $\tilde{t}_0''' = \tilde{q}_0\tilde{a}_{11} - \tilde{q}_1\tilde{a}_{01}, \tilde{c}'' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$ $\tilde{d} = \tilde{d}' -' \tilde{c}'', i = \tilde{d}^{-1}, \tilde{c} = \tilde{c}''i, \tilde{s}_0'' = (\tilde{t}_0''' - s_0''\tilde{c}'')i$ if $\tilde{d} = 0$ or $\tilde{c} = 0$ then different case $Z_{42} = Z_{32}\tilde{c}, z_{42} = Z_{42}^2$	I+12M+1S
6	Precomputations $\tilde{V}_1 = \tilde{v}_{11}x + \tilde{v}_{10}$ $\tilde{v}_{11} = \bar{v}_{11}\tilde{c} (= v_{11}Z_{42}), \tilde{v}_{10} = \bar{v}_{10}\tilde{c} (= v_{10}Z_{42}0$	2M
7	Compute $U_4 = (S(L + 2V_1) - K)/U_3 = X^2 + u_{41}X + u_{40}$: $u_{40} = \tilde{s}_0''(\tilde{s}_0'' + 2\tilde{a}_{10}) - \tilde{q}_1 + 2\tilde{v}_{11} + (u_{11} + u_{31})z_{42}$ $u_{41} = 2\tilde{s}_0'' + u_{11} - u_{31} - z_{42}$	2M
8	Compute $\bar{V}_4 = Z_{42}V_4$ $\bar{v}_{41} = (u_{41} - \tilde{s}_0'')(u_{11} - u_{41}) + (u_{40} - u_{10}) - \tilde{v}_{11}$ $\bar{v}_{40} = \tilde{s}_0''(u_{40} - u_{10}) + (u_{11} - u_{41})u_{40} - \tilde{v}_{10}$	3M
	Total Cost 2I+36M+2S	

Table A.5: Double-Add with One Inversion for Semi-Affine Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12}, $z_{12} = Z_{12}^2, (U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor reduced divisor $D_2 = (U_2, V_2)$ $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}X + \bar{v}_{10}$ $U_2 = X^2 + u_{21}X + u_{20}$,$V_1 = v_{21}X + v_{20}$	
	Output: $(U_4, \bar{V}_4 = Z_{42}V_4)$, Z_{42} and $z_{42} = Z_{42}^2$ where $D_4 = (U_4, V_4) = 2D_1 + D_2$ is a reduced Divisor $U_4 = X^2 + u_{41}X + u_{40}$, $\bar{V}_4 = \bar{v}_{41}X + \bar{v}_{40}$	
	Expression	Cost
1	Compute $\bar{V}_2 = Z_{12}V_2$ $\bar{v}_{21} = Z_{12}v_{21}$, $\bar{v}_{20} = Z_{12}v_{20}$	2M
2	Compute $S'' = d \times (\bar{V}_2 - \bar{V}_1)/(s_1U_1) \pmod{U_2}$ and $Z_2' = d/s_1$ $a_{00} = u_{10} - u_{20}, a_{01} = \bar{v}_{10} - \bar{v}_{20}$,$a_{10} = u_{11} - u_{21}, a_{11} = \bar{v}_{11} - \bar{v}_{21}$ $d = a_{00}a_{11} - a_{01}a_{10}$, if $d = 0$ then different case $W_0 = u_{21}(u_{11} - u_{21}), q_0 = u_{20}(u_{11} - u_{21}), q_1 = W_0 + (u_{20} - u_{10})$ $s_0''' = a_{11}q_0 - a_{01}q_1, c' = dc = a_{00}q_1 - a_{10}q_0$ $d' = dc', Z_{32} = c'Z_{12}, z_{32} = Z_{32}^2$ if $c' = 0$ then different case	10M+1S
3	Precomputations $v_{11}' = \bar{v}_{11}d', v_{10}' = \bar{v}_{10}d'$	2M
4	Compute $U_2' = d^2U_2 = d^2X^2 + u_{21}'X + u_{20}'$ $D = d^2, u_{21}' = Du_{21}, u_{20}' = Du_{20}$	2M+1S
5	Compute $U_3' = D(U_3 - U_1) = u_{31}'X + u_{30}'$: $u_{30}' = s_0''^2 + (-u_{21}' + 2W_1)(u_{11} - u_{21}) - u_{20}' + 2v_{11}' + (u_{11} + u_{21})z_{32}$ $W_1 = ds_0'', u_{31}' = 2W_1 - u_{21}' - z_{32}$	3M+1S

Table A.6: Double-Add with One Inversion for Semi-Affine Coordinates-Part 2

6	Computation of $\tilde{S}' = -2V_1'/U_3' \pmod{U_1}$ $\tilde{a}_{00} = u_{30}', \tilde{q}_0 = -2v_{10}', \tilde{w}_0 = -u_{10}u_{31}'$ $\tilde{a}_{10} = u_{31}', \tilde{q}_1 = -2v_{11}', \tilde{w}_1 = -u_{11}u_{31}'$ $\tilde{a}_{01} = \tilde{w}_0, \tilde{a}_{11} = \tilde{w}_1 + u_{30}'$ $\tilde{t}_0' = q_0a_{11} - q_1a_{01}, \tilde{t}_1' = a_{00}q_1 - a_{10}q_0, \tilde{d}' = a_{00}a_{11} - a_{01}a_{10}$ if $\tilde{d}' = 0$ then different case	8M
7	Computation of $\tilde{S} = ((V_3 - V_1)/U_1 \pmod{U'}$ $\tilde{d} = D(\tilde{t}_1' + \tilde{d}'), \tilde{i}_1 = \tilde{d}^{-1}, \tilde{s}_0'' = (D\tilde{t}_0' + \tilde{d}(W_1 - u_{31}'))i_1$ if $\tilde{d} = 0$ then different case $i_2 = i_1(\tilde{t}_1' + \tilde{d}), \tilde{c} = -i_1\tilde{d}', \tilde{W}_2 = i_2u_{31}'$ if $\tilde{c}' = 0$ then different case $\tilde{s}_0'' = \tilde{s}_0'' + \tilde{W}_2, Z_{42} = d\tilde{c}Z_{32}, z_{42} = Z_{42}^2$	I+9M+1S
8	Adjustments $u_{31} = \tilde{W}_2 + u_{11}, u_{30} = i_2u_{30}' + u_{10}$	1M
9	Precomputations $\tilde{V}_1 = \tilde{v}_{11}x + \tilde{v}_{10}$ $\tilde{v}_{11}(= \bar{v}_{11}Z_{32}) = v_{11}\tilde{c}, \tilde{v}_{10}(= \bar{v}_{10}Z_{32}) = v_{10}\tilde{c}$	2M
10	Compute $U_4 = (S(L + 2V_1) - K)/U_3 = X^2 + u_{41}X + u_{40}$: $u_{40} = \tilde{s}_0''(2\tilde{s}_0'' + 2(u_{31} - u_{11})) + u_{31}(u_{31} - u_{11}) + u_{30} - u_{10} +$ $2\tilde{v}_{11} + (u_{11} + u_{31})z_{42}$ $u_{41} = 2s_0'' + u_{11} - u_{31} - z_{42}$	3M
11	Compute $\bar{V}_4 = Z_{42}V_4$ $\bar{v}_{41} = (u_{41} - \tilde{s}_0'')(u_{11} - u_{41}) + (u_{40} - u_{10}) - \tilde{v}_{11}$ $\bar{v}_{40} = \tilde{s}_0''(u_{40} - u_{10}) + (u_{11} - u_{31})u_{40} - \tilde{v}_{10}$	3M
	Total Cost I+45M+4S	

Table A.7: Tripling with Two Inversions for Semi-Affine Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12},$z_{12} = Z_{12}^2$, $(U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$	
	Output: $(U_2, \bar{V}_2 = Z_{22}V_2, Z_{22}, z_{22})$ where $D_2 = (U_2, V_2) = 2D_1$ is a reduced divisor $U_2 = X^2 + u_{21}X + u_{20}$,$\bar{V}_2 = \bar{v}_{21}x + \bar{v}_{20}$	
1	Compute $V' = 2\bar{V}_1 \pmod{U_1}$ $v'_{11} = 2\bar{v}_{11}, v'_{10} = 2\bar{v}_{10}$	-
2	Precomputations $W_0 = \bar{v}_{11}^2, W_1 = u_{11}^2$	2S
3	Compute $K' = (z_{12}f - \bar{V}^2)/U_1 \pmod{U_1} = k'_1X + k'_0$ $w_3 = f_3 + W_1, w_4 = 2u_{10}, k_1 = (2W_1 + w_3 - w_4)z_{12}$ $k_0 = (u_{11}(2w_4 - w_3) + f_2)z_{12} - W_0$	3M
4	Compute coefficients and determinant d of the matrix $a_{00} = v'_{10}, a_{01} = -k_0, , a_{10} = v'_{11}, a_{11} = -k_1$ $d = a_{00}a_{11} - a_{01}a_{10}$,if $d = 0$ then different case $i = d^{-1}, q_0 = v'_{11}u_{10}, q_1 = v'_{11}u_{11} - v'_{10}$	1I+4M
5	Compute $S'' \equiv K/(2V_1s_1) \pmod{U_1}$ and Z_{22} $c = 1/Z_{12}s_1 = (a_{00}q_1 - a_{10}q_0)i, Z_{22} = Z_{12}c$ if $c = 0$ then different case $s''_0 = (a_{11}q_0 - a_{01}q_1)i, z_{22} = Z_{22}^2$	7M+1S
6	Precomputations $\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$ $\bar{v}_{11} = \bar{v}_{11}c(= v_{11}Z_{22}), \bar{v}_{10} = \bar{v}_{10}c(= v_{10}Z_{22})$	2M
7	Compute $U_2 = (S(SU_1 + 2V_1) - K)/U_1$ $u_{20} = s''^2_0 + 2\bar{v}_{11} + 2u_1z_{22}$ $u_{21} = 2s''_0 - z_{22}$	1M+1S

Table A.8: Tripling with Two Inversions for Semi-Affine Coordinates-Part 2

8	Compute $S'' = (\tilde{V}_3 - \tilde{V}_1)/(\tilde{s}_1 U_1) \pmod{U_3}$ and $Z_{32} = 1/s_1$ $\tilde{a}_{00} = u_{10} - u_{20}, \tilde{a}_{01} = 2\bar{v}_{10}, \tilde{q}_0 = u_{20}(u_{11} - u_{21})$ $\tilde{a}_{10} = u_{11} - u_{21}, \tilde{a}_{11} = 2\tilde{v}_{11}, \tilde{W}_0 = u_{21}(u_{11} - u_{21}), \tilde{q}_1 = \tilde{W}_1 - \tilde{a}_{00}$ $\tilde{d}' = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}$ $\tilde{t}_0''' = \tilde{q}_0\tilde{a}_{11} - \tilde{q}_1\tilde{a}_{01}, \tilde{c}'' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$ $\tilde{d} = \tilde{d}' -' \tilde{c}'', i = \tilde{d}^{-1}, \tilde{c} = \tilde{c}''i, \tilde{s}_0'' = (\tilde{t}_0''' - s_0''\tilde{c}'')i$ if $\tilde{d} = 0$ or $\tilde{c} = 0$ then different cases. $Z_{32} = Z_{22}\tilde{c}, z_{32} = Z_{32}^2$	I+12M+1S
9	Precomputations $\tilde{V}_1 = \tilde{v}_{11}x + \tilde{v}_{10}$ $\tilde{v}_{11} = \tilde{v}_{11}\tilde{c}(= v_{11}Z_{32}), \tilde{v}_{10} = \tilde{v}_{10}\tilde{c}(= v_{10}Z_{32})$	2M
10	Compute $U_3 = (\tilde{S}(\tilde{S}U_1 + 2V_1) - K)/U_1$ $u_{31} = 2\tilde{s}'' + \tilde{a}_{10} - z_{32}$ $u_{30} = \tilde{s}_0''(\tilde{s}_0'' + 2\tilde{a}_{10}) - \tilde{q}_1 + 2\tilde{v}_{11} + (u_{11} + u_{21})z_{32}$	2M
11	Compute $\tilde{V}_3 = V_3 Z_{32}$ $v_{30} = \tilde{s}_0''(u_{30} - u_{10}) + (u_{11} - u_{31})u_{30} - \tilde{v}_{10}$ $v_{31} = (u_{31} - \tilde{s}_0'')(u_{11} - u_{31}) + (u_{30} - u_{10}) - \tilde{v}_{11}$	3M
	Total Cost 2I+36M+5S	

Table A.9: Tripling with One Inversion for Semi-Affine Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ Z_{12},$z_{12} = Z_{12}^2$, $(U_1, \bar{V}_1 = Z_{12}V_1)$ where $D_1 = (U_1, V_1)$ is a reduced divisor $U_1 = X^2 + u_{11}X + u_{10}$,$\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$	
	Output: $(U_2, \bar{V}_2 = Z_{22}V_2, Z_{22}, z_{22})$ where $D_2 = (U_2, V_2) = 2D_1$ is a reduced divisor $U_2 = X^2 + u_{21}X + u_{20}$,$\bar{V}_2 = \bar{v}_{21}x + \bar{v}_{20}$	
1	Compute $V' = 2\bar{V}_1 \pmod{U_1}$ $v'_{11} = 2\bar{v}_{11}, v'_{10} = 2\bar{v}_{10}$	-
2	Precomputations $W_0 = \bar{v}_{11}^2, W_1 = u_{11}^2$	2S
3	Compute $K = (z_{12}f - \bar{V}^2)/U_1 \pmod{U_1} = k_1X + k_0$ $w_3 = f_3 + W_1, w_4 = 2u_{10}$ $k_1 = (2W_1 + w_3 - w_4)z_{12}$ $k_0 = (u_{11}(2w_4 - w_3) + f_2)z_{12} - W_0$	3M
4	Compute coefficients and determinant d of the matrix : $a_{00} = v'_{10}, a_{01} = -k_0, , a_{10} = v'_{11}, a_{11} = -k_1$ $d = a_{00}a_{11} - a_{01}a_{10}$,if $d = 0$ then different case $q_0 = v'_{11}u_{10}, q_1 = v'_{11}u_{21} - v'_{10}$	4M
5	Compute $S'' = X + s''_0 \equiv (K/(2V_1s_1)) \pmod{U_1}$ and $Z_{22} = 1/s_1$: $c' = d/Z_{12}s_1 = a_{00}q_1 - a_{10}q_0, Z_{22} = c'Z_{12}$ $s'''_0 = a_{11}q_0 - a_{01}q_1, z_{22} = Z_{22}^2$ if $c' = 0$ then different case	5M+1S
6	Precomputations $\bar{V}_1 = \bar{v}_{11}x + \bar{v}_{10}$ $w_5 = dc', \bar{v}_{11} = \bar{v}_{11}w_5, \bar{v}_{10} = \bar{v}_{10}w_5$	3M
7	Compute $U_2 = (S(SU_1 + 2V_1) - K)/U_1 = X^2 + u_{21}X + u_{20}$ $W_6 = ds'''_0, u_{21} = 2W_6 - z_{22}$ $u_{20} = s''''^2_0 + 2\bar{v}_{11} + 2u_{11}z_{22}$	2M+1S
8	Adjust U_1 $D = d^2, u'_{10} = Du_{10}, u'_{11} = Du_{11}$	2M+1S

Table A.10: Tripling with One Inversion for Semi-Affine Coordinates-Part 2

9	Compute $\tilde{S}'' = (-S - 2\tilde{V}_1/U_1)/\tilde{s}_1 \pmod{U_2}$ and $1/\tilde{s}_1$ $\tilde{a}_{00} = u'_{10} - u_{20}, \tilde{a}_{10} = u'_{11} - u_{21}, \tilde{a}_{01} = -\tilde{a}_{10}u_{20}$ $\tilde{W}_0 = \tilde{a}_{10}u_{21}, \tilde{a}_{11} = \tilde{a}_{00}D - \tilde{W}_0, \tilde{q}_0 = -2\bar{v}_{10}, \tilde{q}_1 = -2\bar{v}_{11}$ $\tilde{d}' = (\tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{10}\tilde{a}_{01}), \tilde{W}_1 = (\tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0)D - \tilde{d}', \tilde{d} = D\tilde{W}_1$ $\tilde{s}'''_0 = (\tilde{q}_0\tilde{a}_{11} - \tilde{q}_1\tilde{a}_{01})D - \tilde{d}'W_6$ if $\tilde{d}' = 0$ different case $i_1 = \tilde{d}^{-1}, \tilde{s}''_0 = \tilde{s}'''_0 i_1, \tilde{c} = \tilde{d}' i_1$ if $\tilde{c} = 0$ different case $Z_{32} = d\tilde{c}Z_{22}, z_{32} = Z_{32}^2$	1I+17M+1S
10	Adjust U_2 $i_2 = \tilde{W}_1 i_1$ $u_{20} = i_2 u_{20}, u_{21} = i_2 u_{21}$	3M
11	Adjust V_1 $\tilde{v}_{11} = \tilde{v}_{11}\tilde{c}, \tilde{v}_{10} = \tilde{v}_{10}\tilde{c}$	2M
12	Compute $U_3 = (\tilde{S}(\tilde{S}U_1 + 2V_1) - K)/U_1 = X^2 + u_{31}X + u_{30}$ $u_{31} = 2\tilde{s}''_0 + u_{11} - u_{21} - z_{32}$ $u_{30} = (u_{10} - u_{20}) + 2\tilde{v}_{11} + (u_{11} + u_{21})z_{32} +$ $\tilde{s}''^2_0 + (u_{11} - u_{21})(2\tilde{s}''_0 - u_{21})$	2M+1S
13	Compute $\bar{V}_3 = V_3/Z_{32}$ $v_{30} = \tilde{s}''_0(u_{30} - u_{10}) + (u_{11} - u_{31})u_{30} - \tilde{v}_{10}$ $v_{31} = (u_{31} - \tilde{s}''_0)(u_{11} - u_{31}) + (u_{30} - u_{10}) - \tilde{v}_{11}$	3M
	Total Cost I+46M+7S	

Table A.11: Doubling In Lange's New Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_{11}, z_{11} = Z_{11}^2, Z_{12}, z_{12} = Z_{12}^2)$ A reduced divisor in Lange's new coordinates $U_1 = z_{11}X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$	
	Output:$D_2 = 2D_1$ $D_2 = (U_2, V_2, Z_{21}, z_{21} = Z_{21}^2, Z_{22}, z_{22} = Z_{22}^2)$ A reduced divisor in Lange's new coordinates $U_2 = z_{21}X^2 + u_{21}X + u_{20}$,$V_2 = v_{21}X + v_{20}$	
1	Compute $V_1' = 2V_1$ $v_{11}' = 2v_{11}, v_{10}' = 2v_{10}$	-
2	Compute $K \pmod{U_1}$ $w_0 = v_{11}^2, w_1 = u_{11}^2, W_2 = z_{11}^2, W_3 = u_{10}z_{11}$ $w_4 = f_3W_2 + w_1, w_5 = 2W_3$ $k_1 = (2w_1 + w_4 - w_5)z_{12}, k_0 = (u_{11}(2w_5 - w_4) + f_2W_2z_{11})z_{12} - w_0$	5M+3S
3	Set-up the linear system and solve $a_{00} = v_{10}'z_{11}, a_{10} = v_{11}', a_{01} = -k_0, a_{11} = -k_1$ $q_0 = v_{11}'W_3, q_1 = v_{11}'u_{11} - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - q_0a_{10}$	9M
4	Precomputations $W_6 = c'Z_{12}, W_7 = Z_{11}W_6, W_8 = dz_{11}$ $W_9 = W_8^2, W_{10} = W_7^2, W_{11} = W_8s_0''''$	4M+2S
5	Adjust V_1 $W_{12} = dc', v_{11}' = W_{12}v_{11}, v_{10}' = W_{12}v_{10}$	3M
6	Compute U_2 $u_{21} = 2W_{11} - W_{10}, u_{20} = s_0''''^2 + 2v_{11}' + 2u_{11}W_6^2$	1M+2S
7	Adjust U_1 $W_{13} = W_8d, u_{11}' = W_{13}u_{11}, u_{10}' = W_{13}u_{10}$	3M
8	Compute weights $Z_{21} = W_8, z_{21} = W_9, Z_{22} = W_7, z_{22} = W_{10}$	- -
9	Compute V_2 $v_{21} = (W_{11} - u_{21})(u_{21} - u_{11}') + z_{21}(u_{20} - u_{10} - \tilde{v}_{11})$ $v_{20} = \tilde{W}_{11}(u_{20} - u_{10}') - (u_{11} - u_{11}')u_{20} - z_{21}\tilde{v}_{10}'$	5M
	Total Cost:30M+7S	

Table A.12: Mixed Addition for Lange's New Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_2, V_2, Z_{11}, z_{11} = Z_{11}^2, Z_{12}, z_{12} = Z_{12}^2)$ A reduced divisor in Lange's new coordinates $U_1 = z_{11}X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$ $D_2 = (U_2, V_2)$ A reduced divisor in Affine coordinates $U_2 = X^2 + u_{21}X + u_{20}$,$V_2 = v_{21}X + v_{20}$	
	Output:$D_3 = D_1 + D_2$ $D_3 = (U_3, V_3, Z_{31}, z_{31} = Z_{31}^2, Z_{32}, z_{32} = Z_{32}^2)$ A reduced divisor in Lange's new coordinates $U_3 = z_{31}X^2 + u_{31}X + u_{30}$,$V_3 = v_{31}X + v_{30}$	
1	Precomputations $W_1 = Z_{11}Z_{12}, W_2 = z_{11}W_1$	2M
2	Normalize V_2 with respect to V_1 $v'_{20} = v_{20}W_2, v'_{21} = v_{21}W_2$	2M
3	Normalize U_2 with respect to U_1 $u'_{20} = u_{20}z_{11}, u'_{21} = u_{21}z_{11}$	2M
4	Compute weighted monic form of S $a_{00} = u_{10} - u'_{20}, a_{10} = u_{11} - u'_{21}$ $a_{01} = v_{10} - v'_{20}, a_{11} = v_{11} - v'_{21}$ $q_0 = a_{10}u_{20}, W_3 = a_{10}u_{21} - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_1a_{11} - q_1a_{01}, c' = a_{00}q_1 - a_{10}q_0$ if $d = 0$ or $c' = 0$ then different cases	8M
5	Precomputations $D = d^2, W_4 = dc', W_5 = c'W_1, W_6 = W_5Z_{11}$ $W_7 = W_5^2, W_8 = W_6^2, W_9 = s_0''''z_{11}, W_{10} = dW_9$	5M+3S
6	Adjust V_1 $v'_{11} = W_4v_{11}, v'_{10} = W_4v_{10}$	2M
7	Compute U_3 $u_{31} = 2W_{10} + a_{10}D - W_8$ $u_{30} = s_0''''(W_9 + 2a_{10}d) - Dq_1 + 2v'_{11} + (u_{11} + u'_{21})W_7$	5M
8	Adjust U_1 $u'_{11} = Du_{11}, u'_{10} = Du_{10}$	2M
9	Compute weights $Z_{31} = dZ_{11}, z_{31} = Z_{31}^2, Z_{32} = W_6, z_{32} = W_8$	1M+1S
10	Compute V_3 $v_{31} = (u_{31} - W_{10})(u'_{11} - u_{31}) + z_{31}(u_{31} - u'_{11} - v'_{11})$ $v_{30} = W_{10}(u_{30} - u'_{10}) + (u'_{11} - u_{31})u_{30} - z_{31}v'_{10}$	5M
	Total Cost 34M+4S	

Table A.13: Double-Add for Lange's New Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_2, V_2, Z_{11}, z_{11} = Z_{11}^2, Z_{12}, z_{12} = Z_{12}^2)$ A reduced divisor in Lange's new coordinates $U_1 = z_{11}X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$ $D_2 = (U_2, V_2)$ A reduced divisor in Affine coordinates $U_2 = X^2 + u_{21}X + u_{20}$,$V_2 = v_{21}X + v_{20}$	
	Output:$D_4 = 2D_1 + D_2$ $D_4 = (U_4, V_4, Z_{41}, z_{41} = Z_{41}^2, Z_{42}, z_{42} = Z_{42}^2)$ A reduced divisor in Lange's new coordinates $U_4 = z_{41}X^2 + u_{41}X + u_{40}$,$V_4 = v_{41}X + v_{40}$	
1	Precomputations $W_1 = Z_{11}Z_{12}, W_2 = z_{11}W_1$	2M
2	Normalize V_2 with respect to V_1 $v'_{20} = v_{20}W_2, v'_{21} = v_{21}W_2$	2M
3	Normalize U_2 with respect to U_1 $u'_{20} = u_{20}z_{11}, u'_{21} = u_{21}z_{11}$	2M
4	Compute weighted monic form of S $a_{00} = u_{10} - u'_{20}, a_{10} = u_{11} - u'_{21}$ $a_{01} = v_{10} - v'_{20}, a_{11} = v_{11} - v'_{21}$ $q_0 = a_{10}u_{20}, W_3 = a_{10}u_{21}, q_1 = W_3 - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - a_{10}q_0$ if $d = 0$ or $c' = 0$ then different cases	8M
5	Precomputations $D = d^2, W_4 = dc', W_5 = c'W_1, W_6 = W_5Z_{11}$ $W_7 = W_5^2, W_8 = W_6^2, W_9 = s_0''''z_{11}, W_{10} = dW_9$	5M+3S
6	Adjust V_1 $v'_{11} = W_4v_{11}, v'_{10} = W_4v_{10}$	2M
7	Compute U_3 $u_{31} = 2W_{10} + a_{10}D - W_8$ $u_{30} = s_0''''(W_9 + 2a_{10}d) - Dq_1 + 2v'_{11} + (u_{11} + u'_{21})W_7$	5M
8	Adjust U_1 $u'_{11} = Du_{11}, u'_{10} = Du_{10}$	2M
9	Compute weights $Z_{31} = dZ_{11}, z_{31} = Z_{31}^2$	1M+1S

Table A.14: Double-Add for Lange's New Coordinates-Part 2

10	Compute weighted $\tilde{T} = -2V_1/U_1 \pmod{U_2}$ $\tilde{a}_{00} = u'_{10} - u_{30}, \tilde{a}_{10} = u'_{11} - u_{31}$ $\tilde{q}_0 = -2v'_{10}, \tilde{q}_1 = -2v'_{11}$ $\tilde{a}_{01} = -\tilde{a}_{10}u_{30}, \tilde{a}_{11} = -\tilde{a}_{10}u_{31} + \tilde{a}_{00}z_{31}$ $\tilde{t}'_1 = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0, \tilde{d}' = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{10}\tilde{a}_{01}$ $\tilde{t}_0 = \tilde{q}_0\tilde{a}_{11} - \tilde{q}_1\tilde{a}_{01}$	9M
11	Precomputation $\tilde{s}''''_0 = \tilde{t}_0 z_{31} - W_{10}\tilde{d}'$ $\tilde{W}_0 = \tilde{d}' Z_{32}, \tilde{W}_1 = \tilde{W}_0 Z_{31}, \tilde{W}_2 = \tilde{d}\tilde{d}'$ $\tilde{d}'' = \tilde{t}'_1 z_{31} - \tilde{d}', \tilde{d} = \tilde{d}' z_{31}$ $\tilde{W}_3 = \tilde{s}''''_0 \tilde{d}, \tilde{W}_4 = \tilde{W}_1^2$ $\tilde{W}_5 = \tilde{d}''\tilde{d}, \tilde{W}_6 = \tilde{d}''^2, \tilde{W}_7 = \tilde{W}_0^2$	9M+2S
12	Adjust V_1 $v'_{11} = v'_{11}\tilde{W}_2, \tilde{v}'_{10} = v'_{10}\tilde{W}_2$	2M
13	Compute U_4 $u_{41} = 2\tilde{W}_3 - \tilde{W}_4 + \tilde{W}_5\tilde{a}_{10}$ $u_{40} = \tilde{s}''''_0(\tilde{s}''''_0 + 2\tilde{a}_{10}\tilde{d}'') + \tilde{a}_{11}\tilde{W}_6 + 2\tilde{v}'_{11} + (u_{31} + u'_{11})\tilde{W}_7$	5M
15	Compute weights $Z_{41} = \tilde{d}, z_{41} = Z_{41}^2, Z_{42} = \tilde{W}_1, z_{42} = \tilde{W}_4$	1S
16	Adjust U_1 $\tilde{u}'_{11} = u'_{11}\tilde{W}_5, \tilde{u}'_{10} = u'_{10}\tilde{W}_5$	2M
17	Compute V_4 $v_{41} = (u_{41} - \tilde{W}_3)(\tilde{u}'_{11} - u_{41}) + z_{41}(u_{40} - \tilde{u}'_{10} - \tilde{v}'_{11})$ $v_{40} = \tilde{W}_3(u_{40} - \tilde{u}'_{10}) + (\tilde{u}'_{11} - u_{41})u_{40} - z_{41}\tilde{v}'_{10}$	5M
	Total Cost 61M+8S	

Table A.15: Tripling for Lange's New Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_{11}, z_{11} = Z_{11}^2, Z_{12}, z_{12} = Z_{12}^2)$ A reduced divisor in Lange's new coordinates $U_1 = z_{11}X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$	
	Output:$D_3 = 3D_1$ $D_3 = (U_3, V_3, Z_{31}, z_{31} = Z_{31}^2, Z_{32}, z_{32} = Z_{32}^2)$ A reduced divisor in Lange's new coordinates $U_3 = z_{31}X^2 + u_{31}X + u_{30}$,$V_3 = v_{31}X + v_{30}$	
1	Compute $V_1' = 2V_1$ $v_{11}' = 2v_{11}, v_{10}' = 2v_{10}$	-
2	Compute $K \pmod{U_1}$ $w_0 = v_{11}^2, w_1 = u_{11}^2, W_2 = z_{11}^2, W_3 = u_{10}z_{11}$ $w_4 = f_3W_2 + w_1, w_5 = 2W_3$ $k_1 = (2w_1 + w_4 - w_5)z_{12}$ $k_0 = (u_{11}(2w_5 - w_4) + f_2W_2z_{11})z_{12} - w_0$	5M+3S
3	Set-up the linear system and solve $a_{00} = v_{10}'z_{11}, a_{10} = v_{11}', a_{01} = -k_0, a_{11} = -k_1$ $q_0 = v_{11}'W_3, q_1 = v_{11}'u_{11} - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - q_0a_{10}$ if $d = 0$ or $c' = 0$ then different cases	9M
4	Precomputations $W_6 = c'Z_{12}, W_7 = Z_{11}W_6, W_8 = dz_{11}$ $W_9 = W_8^2, W_{10} = W_7^2, W_{11} = W_8s_0'''$	4M+2S
5	Adjust V_1 $W_{12} = dc', v_{11}' = W_{12}v_{11}, v_{10}' = W_{12}v_{10}$	3M
6	Compute U_2 $u_{21} = 2W_{11} - W_{10}, u_{20} = s_0''''^2 + 2v_{11}' + 2u_{11}W_6^2$	1M+2S
7	Adjust U_1 $W_{13} = W_8d, u_{11}' = W_{13}u_{11}, u_{10}' = W_{13}u_{10}$	3M
8	Compute weight of U_2 $Z_{21} = W_8, z_{21} = W_9, Z_{22} = W_7, z_{22} = W_{10}$	-

Table A.16: Tripling for Lange's New Coordinates-Part 2

9	Compute weighted $\tilde{T} = -2V_1'/U_2 \pmod{U_1}$ $\tilde{a}_{00} = u_{20} - u_{10}', \tilde{a}_{10} = u_{21} - u_{11}', \tilde{a}_{01} = -2v_{10}', \tilde{a}_{11} = -2v_{11}$ $\tilde{w}_0 = \tilde{a}_{00}z_{21}, \tilde{q}_0 = \tilde{a}_{10}u_{10}', \tilde{q}_{11} = \tilde{a}_{10}u_{11}' - \tilde{w}_0$ $\tilde{W}_1 = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}, \tilde{t}_1' = z_{21}\tilde{W}_1, \tilde{t}_0' = \tilde{a}_{11}\tilde{q}_0 - \tilde{a}_{01}\tilde{q}_1$ $\tilde{d}' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$	10M
10	Compute weighted $\tilde{R} = -2V_1'/U_1 \pmod{U_2}$ $\tilde{r}_1' = \tilde{t}_1', \tilde{r}_0' = \tilde{t}_0' + \tilde{W}_1(u_{21} - u_{11}')$	1M
11	Compute weighted $\tilde{S} = (V_2 - V_1)/U_1 \pmod{U_2}$ $\tilde{W}_3 = \tilde{r}_1' - \tilde{d}', \tilde{W}_4 = z_{21}\tilde{r}_0', \tilde{W}_5 = W_{11}\tilde{d}', \tilde{s}_0''' = -\tilde{W}_5 + \tilde{W}_4$	2M
12	Compute weighted $\bar{S} - \tilde{S}$-S $\bar{s}_1 = \tilde{r}_1' - 2\tilde{d}', \bar{s}_0 = -2\tilde{W}_5 + \tilde{W}_4$	-
13	Compute $U_3' \pmod{U_1}$ $\tilde{W}_6 = -\tilde{t}_1'\bar{s}_1', u_{30}' = \bar{s}_0\tilde{t}_0' + \tilde{W}_6u_{10}'$	3M
14	Compute U_3 $\tilde{W}_7 = (\tilde{W}_3)^2$ $u_{30} = u_{30}' + \tilde{W}_7u_{10}'$	1M+1S
15	Computation of Weights $Z_{31} = \tilde{W}_3Z_{21}, z_{31} = Z_{31}^2$ $Z_{32} = \tilde{d}'Z_{22}, z_{31} = Z_{32}^2$	2M+2S
16	Precomputation $\tilde{W}_8 = \tilde{W}_3\tilde{s}_0''', \tilde{W}_9 = \tilde{d}'\tilde{W}_3$	2M
17	Adjust V_1 $\tilde{v}_{11}' = \tilde{W}_9v_{11}', \tilde{v}_{10}' = \tilde{W}_9v_{10}'$	2M
18	Compute u_{31} $u_{31} = 2\tilde{W}_8 + (u_{11}' - u_{21})\tilde{W}_7 - z_{32}, u_{31}' = u_{31} - u_{11}'\tilde{W}_7$	2M
19'	Compute V_3 $v_{31} = -(u_{31} - \tilde{W}_8)u_{31}' + z_{31}(u_{30}' - \tilde{v}_{11})$ $v_{30} = \tilde{W}_8u_{30}' - u_{31}'u_{30} - z_{31}\tilde{v}_{10}'$	5M
	Total Cost 55M+10S	

Table A.17: Doubling for Projective Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_1)$ A reduced divisor in projective coordinates $U_1 = Z_1X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$	
	Output:$D_2 = 2D_1$ $D_2 = (U_2, V_2, Z_2)$ A reduced divisor in projective coordinates $U_2 = Z_2X^2 + u_{21}X + u_{20}$,$V_2 = v_{21}X + v_{20}$	
1	Compute $V_1' = 2V_1$ $v_{11}' = 2v_{11}, v_{10}' = 2v_{10}$	-
2	Compute $K \pmod{U_1}$ $w_0 = v_{11}^2, w_1 = u_{11}^2, W_2 = Z_1^2w_3 = f_2W_2$ $W_4 = u_{10}Z_1, w_5 = f_3W_2 + w_1, w_6 = 2W_4$ $k_1 = 2w_1 + w_5 - w_6, k_0 = u_{11}(2w_5 - w_4) + (w_3 - w_0)Z_1$	3M+3S
3	Set-up the linear system and solve $W_7 = v_{10}Z_1, a_{00} = 2W_7, a_{10} = v_{11}', a_{10} = -k_0, a_{11} = -k_1$ $q_0 = v_{11}'W_4, q_1 = v_{11}'u_{11} - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - q_0a_{10}$ if $d = 0$ or $c' = 0$ then different cases	9M
4	Precomputations $W_8 = ds_0''', w_9 = c'Z_1, W_{10} = c'w_9, W_{11} = dw_9$	4M
5	Adjust V_1 $v_{11}' = W_{11}v_{11}, v_{10}' = W_{11}W_7$	2M
6	Compute U_2 $u_{21} = 2W_8 - W_{10}, u_{20} = s_0''''^2 + 2v_{11}' + 2u_{11}W_{10}$	1M+1S
7	Compute Z_2 $W_{12} = d^2, W_{13} = W_{11}W_2, Z_2 = W_{12}W_{13}$	2M+1S
8	Adjust U_1 $u_{11}' = W_{12}u_{11}, u_{10}' = W_{12}W_4$	2M
9	Compute V_2 $v_{21} = Z_1((u_{21} - W_8)(u_{11}' - u_{21}) + W_{12}(u_{20} - u_{10}' - v_{11}'))$ $v_{20} = W_8(u_{20} - u_{10}') + u_{20}(u_{11}' - u_{21}) - W_{12}v_{10}'$	6M
10	Adjust U_2 $u_{21} = u_{21}W_{11}Z_1, u_{20} = u_{20}W_{11}$	3M
	Total Cost:32M+5S	

Table A.18: Mixed Addition for Projective Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_1)$ A reduced divisor in projective coordinates $U_1 = Z_1X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$ $D_2 = (U_2, V_2)$ A reduced divisor in affine coordinates $U_2 = X^2 + u_{21}X + u_{10}$,$V_2 = v_{21}X + v_{20}$	
	Output:$D_3 = D_1 + D_2$ $D_3 = (U_3, V_3, Z_3)$ A reduced divisor in projective coordinates $U_3 = Z_3X^2 + u_{31}X + u_{30}$,$V_2 = v_{31}X + v_{30}$	
1	Normalize D_2 w.r.t D_1 $u'_{21} = u_{21}Z_1, u'_{20} = u_{20}Z_1, v'_{21} = v_{21}Z_1, v'_{20} = v_{20}Z_1$	4M
2	Set up the linear system and solve $a_{00} = u_{10} - u'_{20}, a_{10} = u_{11} - u'_{21}$ $a_{01} = v_{10} - v'_{20}, a_{11} = v_{11} - v'_{21}$ $q_0 = u_{20}a_{10}, W_0 = a_{10}u_{21}, q_0 = W_0 - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s''''_0 = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - a_{10}q_0$ if $d = 0$ or $c' = 0$ then different cases	8M
3	Precomputation $D = d^2, W_1 = s''''_0Z_1, W_2 = dW_1, W_3 = c'^2, W_4 = W_3Z_1$	3M+2S
4	Adjust V_1 $W_5 = dc'_1, v'_{11} = W_5v_{11}, v'_{10} = W_5v_{10}$	3M
5	Compute U_3 $u_{31} = 2W_2 + a_{10}D - W_4$ $u_{30} = s''''_0(W_1 + 2da_{10}) + D(a_{00} - W_0) + 2v'_{11} + (u'_{21} + u_{11})W_3$	5M
6	Compute Z_3 $W_6 = DZ_1, W_7 = W_5Z_1, Z_3 = W_6W_7$	3M
7	Adjust U_1 $u'_{11} = Du_{11}, u'_{10} = Du_{10}$	2M
8	Compute V_3 $v_{30} = W_2(u_{30} - u'_{10}) + (u'_{11} - u_{31})u_{30} - W_6v'_{10}$ $v_{31} = (u_{31} - W_2)(u'_{11} - u_{31}) + W_6(u_{30} - u'_{10} - v'_{11})$	5M
9	Adjust U_3 $u_{31} = W_7u_{31}, u_{30} = W_7u_{30}$	2M
	Total Cost 35M+2S	

Table A.19: Tripling for Projective Coordinates-Part 1

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_1)$ A reduced divisor in projective coordinates $U_1 = Z_1X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$	
	Output:$D_3 = 3D_1$ $D_3 = (U_3, V_3, Z_3)$ A reduced divisor in projective coordinates $U_3 = Z_3X^2 + u_{31}X + u_{30}$,$V_3 = v_{31}X + v_{30}$	
1	Compute $V_1' = 2V_1$ $v_{11}' = 2v_{11}, v_{10}' = 2v_{10}$	-
2	Compute $K \pmod{U_1}$ $w_0 = v_{11}^2, w_1 = u_{11}^2, W_2 = Z_1^2, w_3 = f_2W_2, W_4 = u_{10}Z_1$ $w_5 = f_3W_2 + w_1, w_6 = 2W_4$ $k_1 = 2w_1 + w_5 - w_6, k_0 = u_{11}(2w_6 - w_5) + (w_3 - w_0)Z_1$	3M+3S
3	Set-up the linear system and solve $a_{00} = v_{10}'Z_1, a_{10} = v_{11}', a_{10} = -k_0, a_{11} = -k_1$ $q_0 = v_{11}'W_4, w_7 = v_{11}'u_{11}, q_1 = w_7 - a_{00}$ $d = a_{00}a_{11} - a_{01}a_{10}, s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - q_0a_{10}$ if $d = 0$ or $c' = 0$ then different cases	9M
4	Precomputations $Z_{21} = dZ_1, z_{21} = Z_{21}^2$ $W_8 = Z_{21}s_0'''', w_9 = c'Z_1, W_{10} = c'w_9$ $W_{11} = dw_9, Z_{22} = w_9, z_{22} = Z_{22}^2$	5M+2S
5	Adjust V_1 $v_{11}' = W_{11}v_{11}, v_{10}' = W_{11}v_{10}$	2M
6	Compute U_2 $u_{21} = 2W_8 - z_{22}, u_{20} = s_0''''^2 + 2v_{11}' + 2u_{11}W_{10}$	2M+1S
7	Adjust U_1 $w_{12} = dZ_{21}, u_{11}' = w_{12}u_{11}, u_{10}' = w_{12}u_{10}$	2M

Table A.20: Tripling for Projective Coordinates-Part 2

8	Compute weighted $\tilde{T} = -2V_1'/U_2 \pmod{U_1}$ $\tilde{a}_{00} = u_{20} - u_{10}', \tilde{a}_{10} = u_{21} - u_{11}', \tilde{a}_{01} = -2v_{10}', \tilde{a}_{11} = -2v_{11}$ $\tilde{w}_0 = \tilde{a}_{00}z_{21}, \tilde{q}_0 = \tilde{a}_{10}u_{10}', \tilde{q}_{11} = \tilde{a}_{10}u_{11}' - \tilde{w}_0$ $\tilde{W}_1 = \tilde{a}_{00}\tilde{a}_{11} - \tilde{a}_{01}\tilde{a}_{10}, \tilde{t}_1' = z_{21}\tilde{W}_1, \tilde{t}_0' = \tilde{a}_{11}\tilde{q}_0 - \tilde{a}_{01}\tilde{q}_1$ $\tilde{d}' = \tilde{a}_{00}\tilde{q}_1 - \tilde{a}_{10}\tilde{q}_0$	10M
9	Compute weighted $\tilde{R} = -2V_1'/U_1 \pmod{U_2}$ $\tilde{r}_1' = \tilde{t}_1', \tilde{r}_0' = \tilde{t}_0' + \tilde{W}_1(u_{21} - u_{11}')$	1M
10	Compute weighted $\tilde{S} = (V_2 - V_1)/U_1 \pmod{U_2}$ $\tilde{W}_3 = \tilde{r}_1' - \tilde{d}', \tilde{W}_4 = z_{21}\tilde{r}_0', \tilde{W}_5 = W_8\tilde{d}', \tilde{s}_0''' = -\tilde{W}_5 + \tilde{W}_4$	2M
11	Compute weighted $\bar{S} - \tilde{S}$-S $\bar{s}_1 = \tilde{r}_1' - 2\tilde{d}', \bar{s}_0 = -2\tilde{W}_5 + \tilde{W}_4$	-
12	Compute $U_3' \pmod{U_1}$ $\tilde{W}_6 = -\tilde{t}_1'\bar{s}_1', u_{30}' = \bar{s}_0\tilde{t}_0' + \tilde{W}_6u_{10}'$	3M
13	Compute U_3 $\tilde{W}_7 = (\tilde{W}_3)^2$ $u_{30} = u_{30}' + \tilde{W}_7u_{10}'$	1M+1S
14	Computation of Weights $Z_{31} = \tilde{W}_3Z_{21}, z_{31} = Z_{31}^2$ $Z_{32} = \tilde{d}'Z_{22}, z_{31} = Z_{32}^2$	2M+2S
15	Precomputation $\tilde{W}_8 = \tilde{W}_3\tilde{s}_0''', \tilde{W}_9 = \tilde{d}'\tilde{W}_3$	2M
16	Adjust V_1 $\tilde{v}_{11}' = \tilde{W}_9v_{11}', \tilde{v}_{10}' = \tilde{W}_9v_{10}'$	2M
17	Compute u_{31} $u_{31} = 2\tilde{W}_8 + (u_{11}' - u_{21})\tilde{W}_7 - z_{32}, u_{31}' = u_{31} - u_{11}'\tilde{W}_7$	2M
18	Compute V_3 $v_{31} = -(u_{31} - \tilde{W}_8)u_{31}' + z_{31}(u_{30}' - \tilde{v}_{11})$ $v_{30} = \tilde{W}_8u_{30}' - u_{31}'u_{30} - z_{31}\tilde{v}_{10}'$	5M
19	Adjust U_3 and compute Z_3 $\tilde{w}_{10} = Z_{31}Z_{32}, Z_3 := z_{31}\tilde{w}_{10}$ $u_{31} = u_{31}\tilde{w}_{10}, u_{30} = u_{30}\tilde{w}_{10}$	4M
	Total Cost 57M+9S	

Table A.21: Special Addition for Projective Coordinates

	Input: Genus 2 HEC C:$Y^2 = f(X)$ $f = X^5 + f_3X^3 + f_2X^2 + f_1X + f_0$ $D_1 = (U_1, V_1, Z_1)$ A reduced divisor in projective coordinates $U_1 = Z_1X^2 + u_{11}X + u_{10}$,$V_1 = v_{11}X + v_{10}$ $D_2 = (U_2, V_2, Z_1)$ A reduced divisor in projective coordinates $U_2 = Z_1X^2 + u_{21}X + u_{10}$,$V_2 = v_{21}X + v_{20}$	
	Output:$D_3 = D_1 + D_2$ $D_3 = (U_3, V_3, Z_3)$and Z_3' A reduced divisor in projective coordinates $U_3 = Z_3X^2 + u_{31}X + u_{30}$,$V_2 = v_{31}x + v_{30}$, $Z_3 = Z_1Z_3'$ $D_1' = (U_1', V_1', Z_1')$ A reduced divisor D_1 in projective coordinates with $Z_1' = Z_3$ $U_1' = Z_1'X^2 + u_{11}'X + u_{10}'$,$V_1' = v_{11}'x + v_{10}'$	
1	Normalize D_2 w.r.t D_1	-
2	Set up the linear system and solve $a_{00} = u_{10} - u_{20}, a_{10} = u_{11} - u_{21}, a_{01} = v_{10} - v_{20}, a_{11} = v_{11} - v_{21}$ $q_0 = u_{20}a_{10}, W_0 = a_{10}u_{21}, q_0 = W_0 - a_{00}Z_1$ $d' = a_{00}a_{11} - a_{01}a_{10}, d = dZ_1$ $s_0'''' = q_0a_{11} - q_1a_{01}, c' = a_{00}q_1 - a_{10}q_0$ if $d' = 0$ or $c' = 0$ then different cases	10M
3	Precomputation $D = d^2, W_1 = s_0''''Z, W_2 = dW_1, W_3 = c'^2, W_4 = W_3Z$	3M+2S
4	Adjust V_1 $W_5 = dc_1', v_{11}' = W_5v_{11}, v_{10}' = W_5v_{10}$	3M
5	Compute U_3 $u_{31} = 2W_2 + a_{10}D - W_4$ $u_{30} = s_0''''(W_1 + 2da_{10}) - dd'q_1 + 2v_{11}' + (u_{21}' + u_{11})W_3$	6M
6	Compute Z_3, Z_3' $W_6 = DZ_1, W_7 = W_5Z_1, Z_3 = W_6W_7, Z_3' = DW_7$	4M
7	Adjust U_1 $u_{11}' = Du_{11}, u_{10}' = Du_{10}$	2M
8	Adjust V_1 $v_{11}' = v_{11}'W_6, v_{10}' = v_{11}'W_6$	2M
9	Compute V_3 $v_{30} = W_2(u_{30} - u_{10}') + (u_{11}' - u_{31})u_{30} - v_{10}'$ $v_{31} = (u_{31} - W_2)(u_{11}' - u_{31}) + W_6(u_{30} - u_{10}') - v_{11}'$	4M
10	Adjust U_3, U_1 $u_{31} = W_7u_{31}, u_{30} = W_7u_{30}, u_{11}' = u_{11}'W_7, u_{10}' = u_{10}'W_7$	4M
	Total Cost 38M+2S	

Bibliography

[1] Roberto M. Avanzi. A note on signed sliding window recoding and left-to-right analogue. In *Selected Areas in Cryptography-SAC 2004*, volume 3357 of *Lecture Notes In Computer Science*, pages 130–143. Springer Verlag, 2005.

[2] Andrew D. Booth. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*, 4:236–240, 1951.

[3] Wieb Bosma. Signed bits and fast exponentiation. *Journal de Théorie des Nombres de Bordeaux*, 13:27–41, 2001.

[4] Wieb Bosma, John Cannon, and Catherine Playoust. The magma algebra system. I. the user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.

[5] Alfred Brauer. On addition chains. *Bulletin of American Mathematical Society*, 45:736–739, 1939.

[6] David Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of Computation*, 48:95–101, 1987.

[7] Henri Cohen. A course in computational algebraic number theory, 2000.

[8] Henri Cohen. Analysis of the flexible window powering algorithm. *Journal of Cryptology*, 18:63–76, 2005.

[9] Henri Cohen and Gerhard Frey (Eds.). Handbook of elliptic and hyperelliptic curve cryptography, 2006.

[10] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation. In *Information and Communication Security-ICICS 1997*, volume 1334 of *Lecture Notes In Computer Science*, pages 282–290. Springer, 1997.

[11] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology-ASIACRYPT'98*, volume 1514 of *Lecture Notes In Computer Science*, pages 51–65. Springer-Verlag, 1998.

[12] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions On Information Theory*, 22:644–654, 1976.

[13] Kirsten Eisenträger, Kristin Lauter, and Peter L. Montgomery. Fast elliptic curve arithmetic and improved weil pairing evaluation. In *Topics in Cryptology (CT-RSA 2003)*, volume 2612 of *Lecture Notes In Computer Science*, pages 343–354. Springer-Verlag, 2003.

[14] Xinxin Fan and Guang Gong. Efficient explicit formulae for genus 2 hyperelliptic curves over prime fields and their implementations. In *Selected Areas in Cryptography-SAC 2007*, volume 4876 of *Lecture Notes In Computer Science*, pages 155–172. Springer-Verlag, 2007.

[15] Pierrick Gaudry and Robert Harley. Counting points on hyperelliptic curves over finite fields. In *Algorithmic Number Theory-ANTS IV*, volume 1838 of *Lecture Notes In Computer Science*, pages 313–332. Springer-Verlag, 2000.

[16] Pierrick Gaudry and Emmanuel Thomé. The $mp\mathbb{F}_q$ library and implementing curve-based key exchanges. In *Software Performance Enhancement for Encryption and Decryption-SPEED 2007*, pages 49–64. ECRYPT - European Network of Excellence for Cryptology, 2007.

[17] Robert Harley. adding.text and doubling.c at http://cristal.inria.fr/~harley/hyper, 2000.

[18] Marc Joye and Sung-Ming Yen. Optimal left-to-right binary signed-digit recoding. *IEEE Transactions on Computers*, 49:740–748, 2000.

[19] Anatoly A. Karatsuba and Yu. P. Ofman. Multiplication of multiplace numbers on automata. *Doklady Akademii Nauk SSSR*, 145:293–294, 1962.

[20] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[21] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

[22] Tanja Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 15:295–328, 2003.

[23] Patrick Longa and Ali Miri. New composite operations and precomputation scheme for elliptic curve cryptosystems over prime fields. In *Public Key Cryptography-PKC 2008*, volume 4939 of *Lecture Notes In Computer Science*, pages 229–247. Springer Verlag, 2008.

[24] Patrick Longa and Ali Miri. New multibase non-adjacent form scalar multiplication and its application to elliptic curve cryptosystems. In *http://eprint.iacr.org/2008/052*, volume 2008/052. Cryptology ePrint Archive, 2008.

[25] Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato. An elementary introduction to hyperelliptic curve. In *http://www.cacr.math.uwaterloo.ca*, volume CORR 2004-08. Centre for Applied Cryptographic Research (CACR) Technical Reports, 2004.

[26] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology-CRYPTO 85*, volume 218 of *Lecture Notes In Computer Science*, pages 417–426. Springer-Verlag, 1985.

[27] Li Ming and Ali Miri. Private communications.

[28] Peter L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[29] Francois Morain and Jorge Olivos. Speeding up the computations on elliptic curve using addition-subtraction chains. *Theoretical Informatics and Applications*, 24:531–543, 1990.

[30] James A. Muir and Douglas R. Stinson. Minimality and other properties of the width-w nonadjacent form. *Mathematics of Computation*, 75:369–384, 2006.

[31] David Mumford. *Tata Lectures on Theta II*, volume 43 of *Progress in Mathematics*. Birkhäuser, 1984.

[32] George W. Reitwiesner. Binary arithmetic. *Advances in Computing*, I:231–308, 1960.

[33] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[34] I. Shafarevich. *Basic Algebraic Geometry*. Springer-Verlag, 1974.

[35] Thomas Wollinger, Jan Pelzl, and Christof Paar. Cantor versus Harley: Optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems. *IEEE Transactions on Computers*, 54:861–872, 2005.