



YAŞAR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
MASTER THESIS

**FASTER POINT ADDITION FORMULAS FOR
HUFF FORM OF ELLIPTIC CURVES**

NERIMAN GAMZE ORHON
THESIS ADVISOR: ASST.PROF. HÜSEYİN HIŞİL

COMPUTER ENGINEERING

PRESENTATION DATE: 21.08.2017

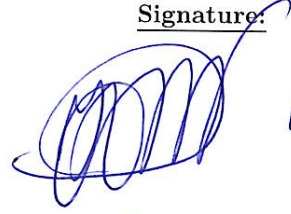
BORNOVA / İZMİR
SEPTEMBER 2017

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members:

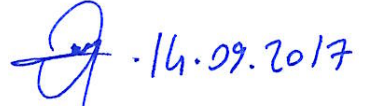
Prof. Urfat NURİYEV, Ph.D.
Ege University

Signature:



14.09.2017

Asst.Prof. Hüseyin HIŞIL, Ph.D.
Yaşar University

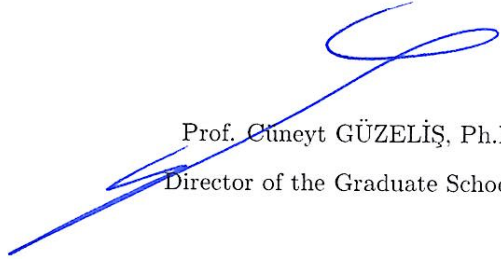


14.09.2017

Asst.Prof. Dindar Öz, Ph.D.
Yaşar University



14.09.2017



Prof. Cüneyt GÜZELİŞ, Ph.D
Director of the Graduate School

ABSTRACT

FASTER POINT ADDITION FORMULAS FOR HUFF FORM OF ELLIPTIC CURVES

Orhon, Neriman Gamze

Msc, Computer Engineering

Advisor: Asst.Prof. Hüseyin HIŞIL, Ph.D.

September 2017

Elliptic curves were being used only for mathematical studies until Miller and Koblitz introduced elliptic curves to crypto-community in 1985 with independent works. Since then, elliptic curves became one of the most significant tools in cryptography. Elliptic curve cryptography (ECC) started to be used for commercial purposes after 1990's. It provides a better level of security with the same key size than the widely used public key crypto-systems such as RSA. Nevertheless, time complexity is not at the desired stage. Hence, there have been several studies so far that aims to increase the time efficiency.

The curve forms that are being used for speed oriented operations came a long way in terms of gathering lower degree formulas for scalar multiplication which is the core operation of ECC. However, one of the curve forms which is called Huff curve could not get competitive with the other forms such as Twisted Edwards, Jacobi Quartic, despite the studies have been made so far. This thesis focuses on increasing the efficiency of Huff form of elliptic curve by making use of mathematical and computational primitives.

Inversion-free point addition and doubling formulas which are being used in scalar multiplication algorithms, are proposed for the Huff curve which is defined as

$$y(1 + ax^2) = cx(1 + dy^2).$$

First idea is rather to embed the curve into a different projective space than the preferred for Huff curve previously. Thus, $\mathbb{P}^1 \times \mathbb{P}^1$ embedding is used instead of \mathbb{P}^2 embedding. The second idea is to make the use of isogenies in order to obtain an alternative doubling formula. Thanks to these two ideas, an improvement is achieved.

The best algorithm for point doubling on Huff curve was computed with $6\mathbf{M} + 5\mathbf{S}$.¹ The proposed doubling formula in this thesis can be computed with $8\mathbf{M}$. Also, operation count of mixed addition is decreased from $10\mathbf{M}$ to $8\mathbf{M}$. Both sets of formulas are leading to an effective cost of $2\mathbf{M}$. Furthermore, they are shown to be 4-way parallel.

Key Words: Elliptic curves, 2-isogeny, efficient, scalar multiplication, Huff curves, inversion-free point addition, parallel computation.

¹ \mathbf{I} , \mathbf{M} , \mathbf{S} , \mathbf{D} , \mathbf{a} represents the cost of various field operations. \mathbf{I} : inversion, \mathbf{M} : multiplication, \mathbf{S} : squaring, \mathbf{D} : multiplication by a curve constant, \mathbf{a} : addition/subtraction.

ÖZ

HUFF ELİPTİK EĞRİ MODELİ ÜZERİNDE HIZLI NOKTA TOPLAMA FORMÜLLERİ

Orhon, Neriman Gamze

Yüksek Lisans Tezi, Bilgisayar Mühendisliği

Danışman: Yrd.Doç.Dr. Hüseyin HIŞIL

Eylül 2017

Önceden yalnızca matematiksel amaçlar için kullanılan eliptik eğriler, 1985 yılında Miller ve Koblitz'in yayınladığı ayrı çalışmalar sayesinde kripto dünyasına giriş yaptı. Bu gelişme ile birlikte eliptik eğriler kriptografinin en önemli araçlarından biri haline geldi. 1990'lardan sonra eliptik eğri tabanlı kriptografi ticari amaçlar için kullanılmaya başladı. Eliptik eğri tabanlı kriptosistemler, RSA gibi sıkça kullanılan asimetrik kriptosistemlerin sağladığı güvenlik seviyesini, daha kısa anahtarlar ile sağlayabildiği için, bu denli hızlı bir gelişim gösterdi. Fakat, sağladığı yüksek seviyeli güvenliğe karşın, verimlilik konusunda yol kat etmesi gerekmektedir. Bu nedenle, eliptik eğri tabanlı kriptosistemlerin verimliliğini arttırmak üzere bir çok çalışma yapılıyor.

Hız amaçlı kullanılan eğri modelleri, eliptik eğri tabanlı kriptografinin temel işlemi olan, skalar çarpım işlemi için kullanılacak, daha düşük dereceli formüller elde edilmesi konusunda büyük yol kat etti. Fakat, bu eğri modellerinden sayılan Huff eğrisi bu vakte kadar yapılan bir çok çalışmaya rağmen, twisted Edwards, Jacobi Quartic gibi eğrilerle rekabet içinde olmadı. Bu tezin amacı, matematiksel ve bilgisayarlı temelleri kullanarak Huff eğri modelinin verimliliğini arttırmaktır.

Bu amaç çerçevesinde,

$$y(1 + ax^2) = cx(1 + dy^2).$$

olarak ifade edilen Huff eğrisi üzerinde, bölmesiz nokta toplama ve nokta çiftleme işlemleri önerildi. Bu amaca ulaşmak için kullanılan ilk yöntem, bölmesiz nokta toplama ve nokta çiftleme işlemleri elde edebilmek için, bu eğri modelinde daha önceki çalışmalarda tercih edilenden başka bir projektif uzay kullanılmasıdır.

İkinci yöntem ise, alternatif bir nokta çiftleme formülü elde etmek için isogenilerden faydalanmaktır. Bu iki yöntem sayesinde, dikkate değer bir gelişim kaydedilmiştir.

Huff eğrisi üzerinde, bilinen en hızlı nokta çiftleme formülü ile, işlem $6M + 5S$ 'de ² hesaplanabiliyordu. Bu tezde sunulan alternatif nokta çiftleme formülü sayesinde $8M$ 'de hesaplanabiliyor. Nokta toplama formülünün işlem sayısı da $10M$ 'den $8M$ 'ye düşürüldü. Bu iki formülde de $2M$ 'lik hızlanma sağlanmıştır. Ayrıca sunulan her iki formül de 4-yönlü paralel olarak işlenebilmektedir.

Anahtar Kelimeler: Eliptik eğriler, 2-isogeny, verimli, skalar çarpım, Huff eğrisi, ters almasız nokta toplama işlemi, paralel hesaplama.

²**I, M, S, D, a** çeşitli cisim işlemlerini temsil eder. **I**: inversion(ters alma), **M**: multiplication(çarpım), **S**: squaring (kare alma), **D**: multiplication by a curve constant (eğri sabiti ile çarpım), **a**: addition/subtraction (toplama/çıkarma).

ACKNOWLEDGEMENTS

First of all, I would like to express my deep gratitude to my advisor Huseyin Hisil for his endless patience and support. Since the second year of my bachelor education, he has never stopped helping me for my research. He has always been a patient mentor for both my academic journey and personal life. Without his encouragement and mentorship, it would be impossible for me to achieve my goal.

I am grateful to organization committees of EUROCRYPT 2017 and Summer School on real-world crypto and privacy. They support me financially to attend their events. Also, they gave me chance to join the community and make presentation of my studies in these events. Their support is magnificently important for the road that I would like to pursue.

I would like to thank Yasar University Scientific Research Project Committee for accepting our project (SRP024) with my advisor Huseyin Hisil and providing me required hardware and software.

I also express my sincere gratitude to Craig Costello. His help about one of the most important parts of my study has improved my thesis. I also would like to thank Peter Schwabe for his advices about my future career. He has always been very friendly and helpful.

Lastly, I would like to thank my loved ones. My parents always supported me and believed in me for everything I did. It would not be possible for me to reach where I stand now, without them. Special thanks to my sister Hande Orhon Ozdag for being my best friend, understanding me all the time and giving me thoughtful advices about my research and personal life. Also, Cenk Ozdag and Mert Can Kilic always encouraged and supported me. I owe much all of them.

Neriman Gamze Orhon

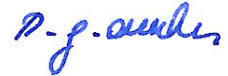
İzmir, 2017



TEXT OF OATH

I declare and honestly confirm that my study, titled “FASTER POINT ADDITION FORMULAS FOR HUFF FORM OF ELLIPTIC CURVES” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Neriman Gamze Orhon



September 14, 2017



PREVIOUSLY PUBLISHED MATERIALS

This thesis contains material based on the following paper. An extended version of paper is submitted to *Designs, Codes and Crypto* Journal, and it is under review.

Orhon, N.G., Hisil, H. (2017). *Speeding up Huff form of elliptic curves*. Cryptology ePrint Archive, Report 2017/320. (<https://eprint.iacr.org/2017/320.pdf>)





TABLE OF CONTENTS

FRONT MATTER	i
ABSTRACT	v
ÖZ	ix
ACKNOWLEDGEMENTS	xiii
TEXT OF OATH	xv
PREVIOUSLY PUBLISHED MATERIALS	xvii
LIST OF FIGURES	xxiii
LIST OF TABLES	xxv
1 INTRODUCTION	1
1.1 MOTIVATION	3
1.2 ROADMAP	4
2 BACKGROUND	7
2.1 ELLIPTIC CURVES	7
2.1.1 SHORT WEIERSTRASS FORM	9
2.1.2 OTHER FORMS OF ELLIPTIC CURVES	11
2.1.3 HUFF CURVE	14
2.2 SCALAR MULTIPLICATION	16
2.3 ISOGENIES	18
2.3.1 VÉLU'S FORMULAS	18
2.3.2 KOHEL'S APPROACH	20

3	EXTENDED HUFF CURVE	21
3.1	DERIVATION OF EXTENDED HUFF CURVE	21
3.2	AFFINE COORDINATES OF EXTENDED HUFF CURVE . . .	23
3.2.1	GROUP LAW IN AFFINE COORDINATES OF EX- TENDED HUFF CURVE	24
3.3	PROJECTIVE COORDINATES OF EXTENDED HUFF CURVE	27
3.3.1	EMBEDDING H INTO \mathbb{P}^2	28
3.3.2	EMBEDDING H INTO $\mathbb{P}^1 \times \mathbb{P}^1$	29
4	2-ISOGENY ON EXTENDED HUFF CURVE	33
4.1	DEFINING ISOGENY WITH KOHEL'S APPROACH	33
4.2	ISOGENY TO EXTENDED HUFF CURVE	35
4.2.1	CONSTRUCTING ISOGENY MAPS BETWEEN 2 DIF- FERENT EXTENDED HUFF CURVES	35
4.2.2	2-ISOGENY IN $\mathbb{P}^1 \times \mathbb{P}^1$	40
4.3	ISOGENY TO TWISTED EDWARDS CURVE	40
4.3.1	CONSTRUCTING ISOGENY MAPS BETWEEN EX- TENDED HUFF CURVE AND TWISTED EDWARDS CURVE	40
4.3.2	2-ISOGENY IN $\mathbb{P}^1 \times \mathbb{P}^1$	45
5	EFFICIENT IMPLEMENTATION	47
5.1	NOT SO FAST ARITHMETIC ON \mathbb{P}^2 EMBEDDING	47
5.1.1	POINT ADDITION ON \mathbb{P}^2 EMBEDDING	47
5.1.2	POINT DOUBLING ON \mathbb{P}^2 EMBEDDING	48
5.2	FASTER ARITHMETIC ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING	49
5.2.1	POINT ADDITION ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING	49
5.2.2	POINT DOUBLING ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING	51
5.2.3	POINT ADDITION AND DOUBLING FOR COFACTOR 4	52

6	COMPARISON AND CONCLUSION	55
6.1	COMPARISON BETWEEN HUFF FORMS	55
6.2	COMPARISON BETWEEN OTHER FORMS OF ELLIPTIC CURVES	57
6.2.1	GROUP OPERATION COSTS FOR DIFFERENT CURVE FORMS	57
6.2.2	SCALAR MUTLIPLICATION COSTS	59
6.2.3	COMPARISON FOR 4-WAY PARALLEL SETTING . . .	61
	REFERENCES	63
	References	63
	APPENDIX 1- C CODE OF SCALAR MULTIPLICATION	66



LIST OF FIGURES

2.1	Short Weierstrass curve $y^2 = x^3 - 10x^2 + 2$ over \mathbb{R}	10
2.2	Huff curve $5x(y^2 - 1) = 20y(x^2 - 1)$ over \mathbb{R}	15
3.1	Huff curve $y(1 + 23x^2) = 10x(1 + 25y^2)$ over \mathbb{R}	24
3.2	Chord and Tangent Rule on extended Huff curve $y(1 + 24x^2) = -3x(1 + 17y^2)$ over \mathbb{R}	25



LIST OF TABLES

6.1	Speed oriented operation counts for Huff form	56
6.2	Operation Counts for Different Curve Forms	58
6.3	Operation Counts with Special Conditions for Different Curve Forms	58
6.4	Cost estimates for 1-NAF variable point scalar multiplication . . .	59
6.5	Cost estimates for sliding window 4-NAF variable point scalar multiplication	60
6.6	Comparison between twisted Edwards and extended Huff curves for 4-way parallel setting	61



CHAPTER 1

INTRODUCTION

Assurance of confidentiality, integrity, non-repudiation and availability of information has always a high importance from the beginning of time. But, the technology era that we live in has the highest dependency on information security since information is transferred through an open network. Thus, much more advanced methods are required now than any other ages. One of the ways that provides the security of information is called *cryptography*.

Cryptography makes use of mathematical techniques in order to transfer information securely through an insecure channel. Thanks to the process of encryption, information becomes unintelligible for parties who does not have the decryption key, and decryption process restores the encrypted message. Until 20th century, *secret key cryptography* is used. In secret key crypto-systems, two parties that want to communicate through an insecure channel use the same secret key, namely encryption and decryption operations are done by a single key. This method is fast but has a remarkable problem. The key distribution must be done in a secure way and both parties must keep the same key secret.

In 1976, Diffie and Hellman (Diffie & Hellman, 1976) came up with a groundbreaking idea which is called *public key cryptography*. This crypto-system is also known as *asymmetric cryptography*, because encryption and decryption operations are done by two different keys. Assume that Bob wants to send a secret message to Alice. Then, Alice generates a key pair. Alice keeps one of the keys secret and sends the other one to Bob. Bob encrypts the message with Alice's public key and sends the encrypted message to Alice. Finally, Alice decrypts the message with the key that she kept secret. In this scenario, key distribution does not cause a problem obviously. However, all public key crypto-systems are slower than the secret key crypto-systems.

After the elliptic curve cryptography (ECC) suggested by the independent works of Miller (Miller, 1985) and Koblitz (Koblitz, 1987), it is started to be used for commercial purposes. It is standardized by (FIPS 186-2, 2000), (ANSI X9.63, 2001), (ISO CD 14888-3, 2006), (IEEE-1363, 2000), (ANSI X9.62, 2005).

This crypto-system is being used for encryption, decryption, key distribution digital signatures given in (Menezes, Van Oorschot, & Vanstone, 1996) and signature verification. Eventhough the elliptic curve cryptography provides a good level of security, it needs to be improved with regards to time efficiency (Hankerson, Menezes, & Vanstone, 2003).

Scalar multiplication, which is composed of addition operations of the points on an elliptic curve, is one of the most common tools of elliptic curve based cryptographic operations. Thus, speeding up point addition, automatically speeds up scalar multiplication and dependent protocols as well.

The following curve forms are the most widely used elliptic curves in speed oriented implementations:

- Short Weierstrass Form: $E_W: y^2 = x^3 + ax + b$
- Extended Jacobi Quartic Form: $E_Q: y^2 = dx^4 + 2ax^2 + 1$
- Jacobi Intersection Form: $E_I: bu^2 + v^2 = 1, au^2 + w^2 = 1$
- Twisted Hessian Form: $E_{He}: ax^3 + y^3 + 1 = dxy$
- Twisted Edwards Form: $E_E: ax^2 + y^2 = 1 + dx^2y^2$

Besides, there is another curve form known as Huff curve which is introduced by Huff in 1948 (Huff, 1948). Recently, this curve form is enrolled to cryptography with the work of Joye, Tibouchi and Vergnaud in (Joye, Tibouchi, & Vergnaud, 2010).

$$E_H: ax(y^2 - 1) = by(x^2 - 1)$$

However, it could not be as efficient as the other forms of elliptic curves in terms of group operations.

1.1 MOTIVATION

The main goal of this thesis is to increase the efficiency of group operations on Huff curve. Group operation formulas include elementary operations such as multiplication, squaring, multiplication by a constant, addition/subtraction and inversion. Apparently, inversion operation is much more costly than others (Hankerson et al., 2003). Therefore, eliminating the inversion operation is the first step of increasing the efficiency of scalar multiplication.

Affine point addition formulas of elliptic curve points involve inversions. But, in the case that the curve is embedded into a projective space, formulas become inversion free.

Definition 1.1.1. Let \mathbb{K} be a field, c and $d \in \mathbb{N}^+$. The following equivalence relation can be defined on $\mathbb{K}^3 - \{(0, 0, 0)\}$:

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2)$$

where $X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2$ for $\exists \lambda \in \mathbb{K}^*$. The equivalence class is $(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in \mathbb{K}^*\}$

- $(X : Y : Z)$ is called projective point,
- $(X/Z^c, Y/Z^d, 1)$ is a representative of the projective point $(X : Y : Z)$,
- The set of all affine points is $\mathbb{A}(\mathbb{K}) = \{(x, y) : x, y \in \mathbb{K}\}$,
- The set of all projective points $\mathbb{P}(\mathbb{K})^0 = \{(X : Y : Z) : X, Y, Z \in \mathbb{K}, Z \neq 0\}$,
- A projective point can be obtained by replacing x by X/Z^c and y by Y/Z^d , (Hankerson et al., 2003).

Embedding any form of elliptic curves into projective space leads to obtain point addition formulas which does not include field inversion. The only inversion is done at the very end of scalar multiplication, while converting projective points to affine ones. If we suppose 256 bit scalar multiplication where the point addition

and doubling operations are done many times, only one inversion that comes from this conversion is unessential. Thus, projective space is necessarily useful in terms of scalar multiplication.

Different projective spaces provide different efficiency for each curve form. More generally Homogeneous Projective Coordinates are being introduced in related literature, nevertheless the efficiency diversity is remarkable. The detailed information can be found in (Hisil, 2010b).

In the previous studies on Huff curve ((Wu & Feng, 2012), (Ciss & Sow, 2011), (Joye et al., 2010)), homogeneous projective coordinates (\mathbb{P}^2 embedding) was used. Unfortunately, this does not provide low degree formulas for Huff curve. So, one of the motivations of this thesis is to try to embed Huff curve into different coordinate system.

The other motivation is to obtain faster formulas for point addition and doubling on Huff curve. It is noticed that elliptic curve isogenies can be used for this purpose.

Isogenies have been used as tool for several elliptic curve based cryptography operations. In 2005, Doche, Icart and Kohel (Doche, Icart, & Kohel, 2006) showed that isogenies can be used in order to increase efficiency of scalar multiplication as well. The composition of a degree 2 isogeny defined between two elliptic curves and its dual gives doubling map of these elliptic curves (Vélu, 1971). This feature of isogenies provides alternative doubling formulas. It is considered that isogenies can also give faster doubling formulas for Huff curves.

The derivation of the several formulas in this work were aided by computer algebra systems MAGMA (Bosma, Cannon, & Playoust, 1997) and Maple (Inc., 2008).

1.2 ROADMAP

This thesis contains 6 chapters including this chapter, and an appendix. It is organized as follows:

- **Chapter 2 - Background** contains background information which is required by the following chapters of this thesis. The main concept of elliptic curves, basic features of different curve models which are used by speed oriented protocols, definition of scalar multiplication and isogenies can be found in this chapter.
- **Chapter 3 - Extended Huff Curves** provides the definition of extended Huff curve equation which is defined in a similar way with the other curve forms. Also, it includes group law properties of extended Huff curve in affine and projective spaces ($\mathbb{P}^1 \times \mathbb{P}^1$ and \mathbb{P}^2).
- **Chapter 4 - 2-isogeny on Extended Huff Curves** provides 2-isogeny maps defined on extended Huff curve. At first, the adoption of Kohel's approach of isogenies to extended Huff curve is explained, and then the construction of 2-isogeny between two extended Huff curves and 2-isogeny between an extended Huff curve and a twisted Edwards curve are introduced in detail. Finally, projective formulas of these isogeny maps are given.
- **Chapter 5 - Efficient Implementation** includes efficient algorithms of the point addition and doubling formulas given in Chapter 3 and Chapter 4. These algorithms are derived for both $\mathbb{P}^1 \times \mathbb{P}^1$ and \mathbb{P}^2 embeddings.
- **Chapter 6 - Comparison and Conclusion** contains comparisons of Huff curve with previous form of Huff curves and other curve forms. Various tables are given in order to observe improvement of extended Huff curve.
- **Appendix A** comprises the C code of 256 bit windowed scalar multiplication on extended Huff curve. The output of this code is the operation counts of the scalar multiplication where the window size is 5.

CHAPTER 2

BACKGROUND

This chapter includes background information which is required by the following chapters of this thesis. Since the main goal of this work is to speed up one of the elliptic curve models in terms of group operations, the definition of elliptic curve is given at first. And then, the curve forms which are used by speed oriented protocols are shown, and their main features such as discriminant and j -invariant values, a brief group law information and projective closures of the each curve form are provided. Huff curve form is covered in more detail than the other forms, because the main focus is to improve its performance. Reader can also find the definition and the algorithm of scalar multiplication of a point on an elliptic curve defined over a finite field, which is used for the implementation phase of this work. Finally, the definition of elliptic curve isogenies is made and the formulas which are required to calculate isogenies between elliptic curves are presented. The following sections make necessary definition calls from ((Cohen et al., 2005), (Hankerson et al., 2003), (Silverman, 2009)).

2.1 ELLIPTIC CURVES

The definition of an elliptic curve defined over finite field \mathbb{K} is given below.

Definition 2.1.1. The general Weierstrass form of elliptic curve E is defined over \mathbb{K} is denoted by:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ together with a single point at infinity.

The curve is non-singular if and only if $\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6 \neq 0$

where

$$b_2 = a_1^2 + 4a_4,$$

$$b_4 = 2a_4 + a_1a_3,$$

$$b_6 = a_3^2 + 4a_6,$$

$$b_8 = a_1^2a_6 - a_1a_3a_4 + 4a_2a_6 + a_2a_3^2 - a_4^2.$$

In the case of $\Delta \neq 0$ the j -invariant is defined as $j(E) = (b_2^2 - 24b_4)^3/\Delta$.

The projective closure of E in \mathbb{P}^2 is given by the equation

$$\mathcal{E}: Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

The point $(X_1 : Y_1 : Z_1)$ is said to be on \mathcal{E} with $X_1, Y_1 \in \mathbb{K}$ and $Z_1 \in \mathbb{K} \setminus \{0\}$. The point at infinity is $\mathcal{O} = (0 : 1 : 0)$. The projective point $(X_1 : Y_1 : Z_1)$ corresponds to affine point $(X_1/Z_1, Y_1/Z_1)$, namely $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1$.

Definition 2.1.2. Let E_1 and E_2 are two elliptic curves defined over field \mathbb{K} such as

$$E_1: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

$$E_2: y^2 + \hat{a}_1xy + \hat{a}_3y = x^3 + \hat{a}_2x^2 + \hat{a}_4x + \hat{a}_6.$$

The curves E_1 and E_2 are *isomorphic* over \mathbb{K} if there exists $c, r, s, t \in \mathbb{K}$ with $c \neq 0$ such that $\hat{a}_1 = (a_1 + 2s)/c$, $\hat{a}_2 = (a_2 - sa_1 + 3r - d^2)/c^2$, $\hat{a}_3 = (a_3 + ra_1 + 2t)/c^3$, $\hat{a}_4 = (a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st)/c^4$, $\hat{a}_6 = (a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1)/c^6$. Then the isomorphism maps between E_1 and E_2 are defined as follows:

$$\phi: E_1 \rightarrow E_2, (x, y) \mapsto \left(\frac{x - r}{c^2}, \frac{y - s(x - r) - t}{c^3} \right),$$

$$\phi^{-1}: E_2 \rightarrow E_1, (x, y) \mapsto (c^2x + r, c^3y + c^2sx + t).$$

2.1.1 SHORT WEIERSTRASS FORM

When $\text{char}(\mathbb{K}) \neq 2, 3$ the following substitutions lead to obtain simpler forms.

Let b_2, b_4, b_6 defined as above and $a, b \in \mathbb{K}$;

- $y \mapsto \frac{1}{2}(y - a_1x - a_3)$ gives

$$E': y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6,$$

- $(x, y) \mapsto \left(\frac{x - 3b_2}{36}, \frac{y}{108} \right)$ gives

$$E_W: y^2 = x^3 + ax + b,$$

which is called *short Weierstrass Form*.

Note that the $\Delta = -16(4a^3 + 27b^2)$ and $j(E_W) = -1728(4a^3)/\Delta$ for short Weierstrass curve. The projective closure of E_W in \mathbb{P}^2 is given by the equation

$$\mathcal{E}_W: Y^2Z = X^3 + aX^2Z + bZ^3.$$

The point $(X_1 : Y_1 : Z_1)$ is said to be on \mathcal{E}_W with $X_1, Y_1 \in \mathbb{K}$ and $Z_1 \in \mathbb{K} \setminus \{0\}$. The point at infinity is $\mathcal{O} = (0 : 1 : 0)$. The projective point $(X_1 : Y_1 : Z_1)$ corresponds to affine point $(X_1/Z_1, Y_1/Z_1)$.

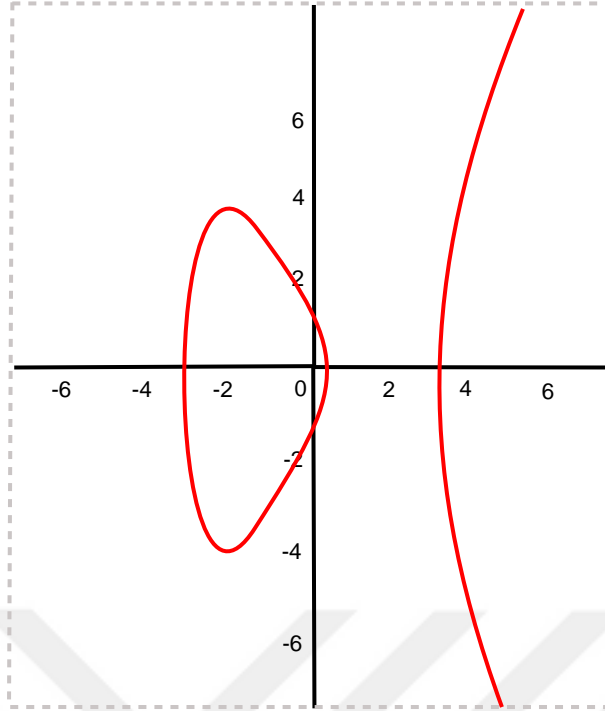


Figure 2.1: Short Weierstrass curve $y^2 = x^3 - 10x^2 + 2$ over \mathbb{R} .

Group Law for Short Weierstrass Form

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on $E_W: y^2 = x^3 + ax + b$

- The negative of P_1 is represented as $-P_1 = (x_1, -y_1)$.
- The double of a point on curve is $[2]P_1 = (x_3, y_3)$

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1,$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 (x_1 - x_3) - y_1$$

where $y_1 \neq 0$ (Silverman, 2009).

- The dedicated addition on E_W is $P_1 + P_2 = (x_3, y_3)$

$$x_3 = \left(\frac{y_1 - y_2}{x_1 - x_2} \right)^2 - x_1 - x_2,$$

$$y_3 = \left(\frac{y_1 - y_2}{x_1 - x_2} \right) (x_1 - x_3) - y_1$$

where $x_1 - x_2 \neq 0$ and $P \neq Q$ (Silverman, 2009).

- The unified addition on E_W is $P_1 + P_2 = (x_3, y_3)$

$$x_3 = \left(\frac{x_1^2 + x_1x_2 + x_2^2 + a}{y_1 + y_2} \right)^2 - x_1 - x_2,$$

$$y_3 = \left(\frac{x_1^2 + x_1x_2 + x_2^2 + a}{y_1 + y_2} \right)(x_1 - x_3) - y_1$$

where $y_1 + y_2 \neq 0$ (Brier & Joye, 2002).

2.1.2 OTHER FORMS OF ELLIPTIC CURVES

In the following subsection, \mathbb{K} denotes a field with the characteristic different from 2 and the constants $a, b, c, d, e, f \in \mathbb{K}$. The content is collected from several resources. In particular, for extended Jacobi quartic form ((Billet & Joye, 2003), (Jacobi, 1829)), for twisted Edwards form ((Bernstein, Birkner, Joye, Lange, & Peters, 2008), (Bernstein & Lange, 2011), (Hisil, Wong, Carter, & Dawson, 2008)), for twisted Hessian ((Bernstein, Chuengsatiansup, Kohel, & Lange, 2015), (Joye & Quisquater, 2001), (Smart, 2001)), for twisted Jacobi intersection ((Feng, Nie, & Wu, 2010)) are studied. Also, Explicit-formulas Database (Bernstein & Lange, 2017) and (Hisil, 2010b) were very helpful for each curve form.

1. **Extended Jacobi Quartic Curve** defined over \mathbb{K} is represented as follows

$$E_Q: y^2 = dx^4 + 2ax^2 + 1.$$

- When $\Delta = d(a^2 - d) \neq 0$, the curve is non singular and $j(E_Q) = \frac{64(a^2 + 3d^2)}{d(a^2 - d)^2} \in \mathbb{K}$.
- Let the point $P = (x_1, y_1)$ on E_Q , the negative of P is $-P = (-x_1, y_1)$.
- E_Q in projective coordinates corresponds to

$$\mathcal{E}_Q: Y^2Z^2 = dX^4 + 2aX^2Z^2 + Z^4.$$

The projective point $(X : Y : Z)$ corresponds to affine point $\left(\frac{X}{Z}, \frac{Y}{Z} \right)$

where $Z \neq 0$. The point $\mathcal{O} = (0: 1: 0)$ is the point at infinity.

- The projective closure of extended Jacobi quartic curve is isomorphic over \mathbb{K} to the Weierstrass curve $Q_W: y^2 = x(x^2 - 4ax - 4a^2 - 4d)$.

Birational maps are given as follows:

$$\begin{aligned}\phi_Q: E_Q &\rightarrow Q_W, (x, y) \mapsto \left(\frac{2y+2}{2x} + 2a, \frac{4y+4}{x^3} + \frac{4a}{x} \right) \\ \phi_Q^{-1}: Q_W &\rightarrow E_Q, (x, y) \mapsto \left(2\frac{x}{y}, 2(x-2a)\frac{x^2}{y^2} - 1 \right).\end{aligned}$$

2. **Twisted Edwards Curve** defined over \mathbb{K} is represented as follows

$$E_E: ax^2 + y^2 = 1 + dx^2y^2.$$

- When $\Delta = ad(a-d) \neq 0$, the curve is non singular and $j(E_E) = \frac{16(a^2 + 14ad + d^2)^3}{ad(a-d)^4} \in \mathbb{K}$.
- Let the point $P = (x_1, y_1)$ on E , the negative of P is $-P = (-x_1, y_1)$.
- E_E in projective coordinates corresponds to

$$\mathcal{E}_E: aX^2Z^2 + Y^2Z^2 = Z^4 + dX^2Y^2.$$

The projective point $(X: Y: Z)$ corresponds to affine point $\left(\frac{X}{Z}, \frac{Y}{Z}\right)$ where $Z \neq 0$. The point $\mathcal{O} = (0: 1: 1)$ is the point at infinity.

- The projective closure of twisted Edwards form of curve is isomorphic over \mathbb{K} to the Weierstrass curve $E_W: y^2 = x^3 + 2(a+d)x^2 + (a-d)^2x$.

Birational maps are given as follows:

$$\begin{aligned}\phi_E: E_E &\rightarrow E_W, (x, y) \mapsto \left((1+y)^2 \frac{1-dx^2}{x^2}, 2(1+y)^2 \frac{1-dx^2}{x^3} + \frac{4a}{x} \right) \\ \phi_E^{-1}: E_W &\rightarrow E_E, (x, y) \mapsto \left(2\frac{x}{y}, \frac{x-a+d}{x+a-d} \right).\end{aligned}$$

3. **Twisted Hessian Curve** defined over \mathbb{K} is represented as follows

$$E_{He}: ax^3 + y^3 + 1 = dxy.$$

- When $\Delta = a(d^3 - 27)^3 \neq 0$, the curve is non singular and $j(E_{He}) = \frac{d^3(d^3 + 216a)^3}{a(d^3 - 27a)^3} \in \mathbb{K}$.
- Let the point $P = (x_1, y_1)$ on E_{He} , the negative of P is $-P = (-x_1/y_1, 1/y_1)$.
- E_{He} in projective coordinates corresponds to

$$\mathcal{E}_H: aX^3 + Y^3 + Z^3 = dXYZ.$$

The projective point $(X : Y : Z)$ corresponds to affine point $\left(\frac{X}{Z}, \frac{Y}{Z}\right)$ where $Z \neq 0$. The point $\mathcal{O} = (0 : -1 : 1)$ is the point at infinity.

- Every twisted Hessian form of curve is isomorphic over \mathbb{K} to the Weierstrass curve $H_W: y^2 = x^3 - \frac{d^4 + 216da}{48}x + \frac{d^6 - 540d^3a - 5832a^2}{864}$.
Birational maps are given as follows:

$$\begin{aligned} \phi_{He}: E_{He} \rightarrow H_W, (x, y) &\mapsto \left(\frac{(d^3 - 27a)x}{3(3 + 3y + dx)} - \frac{d^2}{4}, \frac{(d^3 - 27a)(1 - y)}{23(3 + 3y + dx)} \right) \\ \phi_{He}^{-1}: H_W \rightarrow E_{He}, (x, y) &\mapsto \left(\frac{18d^2 + 72x}{d^3 - 12dx - 108a + 24y}, \right. \\ &\left. 1 - \frac{48y}{d^3 - 12dx - 108a + 24y} \right). \end{aligned}$$

4. **Twisted Jacobi Intersection Curve** defined over \mathbb{K} is represented as follows

$$E_I: bu^2 + v^2 = 1, au^2 + w^2 = 1.$$

- When $\Delta = ab(a - b) \neq 0$, the curve is non singular and $j(E_I) = \frac{256(a^2 - ab + b^2)}{(ab - (a - b))^2} \in \mathbb{K}$.
- Let the point $P = (u_1, v_1, w_1)$ on E_I , the negative of P is $-P = (-U_1, V_1, W_1)$.

- E_I in projective coordinates corresponds to

$$\mathcal{E}_I: bU^2 + V^2 = Z^2, aU^2 + W^2 = Z^2.$$

The projective point $(U: V: W: Z)$ corresponds to affine point $\left(\frac{U}{Z}, \frac{V}{Z}, \frac{W}{Z}\right)$ where $Z \neq 0$. The point $\mathcal{O} = (0: 1: 1: 1)$ is the point at infinity.

- Every twisted Jacobi Intersection form of curve is isomorphic over \mathbb{K} to the Weierstrass curve $I_W: y^2 = x(x-a)(x-b)$. Birational maps are given as follows:

$$\begin{aligned} \phi_I: E_I &\rightarrow I_W, (u, v, w) \mapsto \left(\frac{(1+v)(1+w)}{u^2}, -\frac{(1+v)(1+w)(v+w)}{u^3} \right) \\ \phi_I^{-1}: I_W &\rightarrow E_I, (x, y) \mapsto \left(\frac{2y}{ab-x^2}, 2x\frac{b-x}{ab-x^2} - 1, 2x\frac{a-x}{ab-x^2} - 1 \right). \end{aligned}$$

2.1.3 HUFF CURVE

In this subsection, \mathbb{K} indicates a field of characteristic $\neq 2$ and the constants $a, b \in \mathbb{K}$.

Definition 2.1.3. The Huff curve defined over \mathbb{K} is shown as

$$E_H: ax(y^2 - 1) = by(x^2 - 1)$$

where $ab(a-b) \neq 0$. The j -invariant of Huff curve is $j(E_H) = \frac{2^8(a^4 - a^2b^2 + b^4)^3}{a^4b^4(a^2 - b^2)^2}$.

Let $e, f \in \mathbb{K}$, every Huff curve of form $E_H: ax(y^2 - 1) = by(x^2 - 1)$ is isomorphic over \mathbb{K} to the Weierstrass curve of the form $W_H: y^2 = x(x-e)(x-f)$ where $e = a^2 - b^2$ and $f = a^2$. The change of variables is represented with the

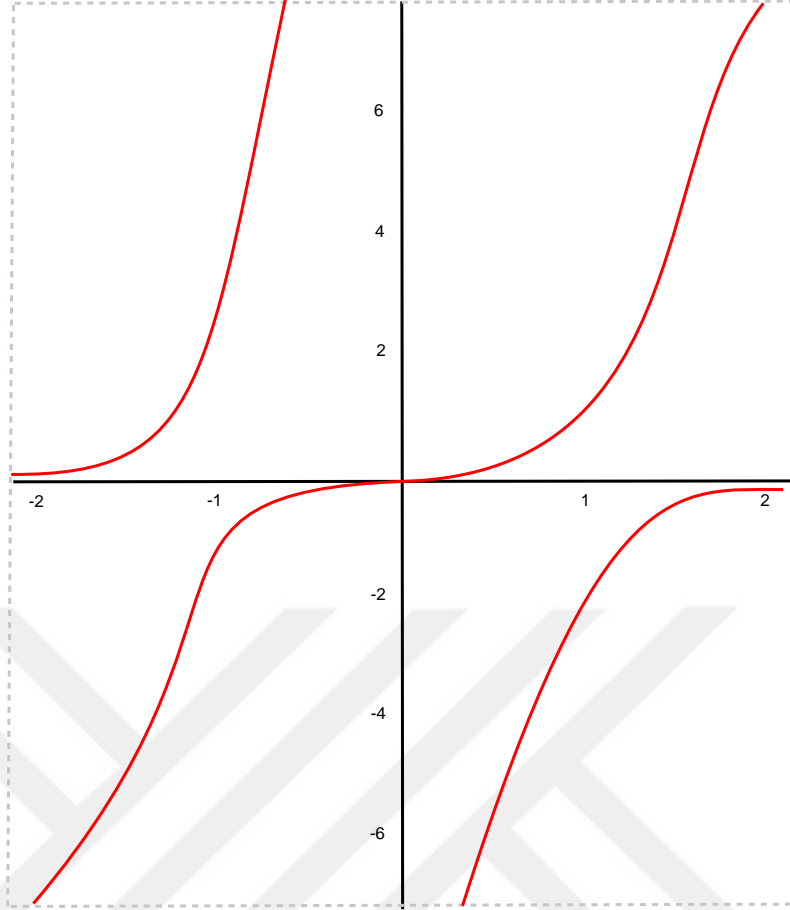


Figure 2.2: Huff curve $5x(y^2 - 1) = 20y(x^2 - 1)$ over \mathbb{R} .

maps below:

$$\begin{aligned} \phi: E_H &\rightarrow H_W, (x, y) \mapsto \left(\frac{a(a + bxy)}{y^2}, \frac{-ab(a^2 - b^2)}{(ax - by)} \right) \\ \phi^{-1}: H_W &\rightarrow E_H, (x, y) \mapsto \left(\frac{-bx}{y}, \frac{-a(x - a^2 + b^2)}{y} \right). \end{aligned}$$

The projective closure of E_H in \mathbb{P}^2 is given by the equation

$$\mathcal{E}_H: aX(Y^2 - Z^2) = bY(X^2 - Z^2).$$

The projective point $(X : Y : Z)$ corresponds to affine point $(X/Z, Y/Z)$.

Group Law on Huff Curve

The identity element of Huff curve is $\mathcal{O} = (0 : 0 : 1)$. There are three points at infinity such that $(1 : 0 : 0), (0 : 1 : 0), (a : b : 0)$. One can obtain the third point at

infinity when they add two of them.

Let P_1, P_2, P_3 be projective points on \mathcal{E}_H . The group operation of Huff curve refers $P_1 + P_2 + P_3 = \mathcal{O}$. The inverse of a point can be found by adding one of the point at infinity to specified point. More specifically, assume that $P_1 = (X_1 : Y_1 : Z_1)$ then the inverse of P_1 is $(X_1 : Y_1 : Z_1) + (0 : 1 : 0) = (X_1 : Y_1 : -Z_1) = -P_1$.

- **dedicated addition:** $P_1 + P_2 = (x_3, y_3) =$

$$\left(\frac{(x_1 - x_2)(y_1 + y_2)}{(y_1 - y_2)(1 - x_1x_2)}, \frac{(x_1 + x_2)(y_1 - y_2)}{(x_1 - x_2)(1 - y_1y_2)} \right) \quad (2.1)$$

where $P_1 \neq P_2$.

- **unified addition:** $P_1 + P_2 = (x_3, y_3) =$

$$\left(\frac{(x_1 + x_2)(1 + y_1 - y_2)}{(1 - y_1y_2)(1 + x_1x_2)}, \frac{(1 + x_1x_2)(y_1 + y_2)}{(1 - x_1x_2)(1 + y_1y_2)} \right). \quad (2.2)$$

- **doubling:** $[2]P_1 = (x_3, y_3) =$

$$\left(\frac{2x_1(1 + y_1^2)}{(y_1^2 - 1)(1 + x_1^2)}, \frac{2y_1(1 + x_1^2)}{(x_1^2 - 1)(1 + y_1^2)} \right). \quad (2.3)$$

2.2 SCALAR MULTIPLICATION

The following assumptions are used for below two definitions.

Let \mathbb{K} be a field, E be an elliptic curve defined over \mathbb{K} , P and Q are points on E and $n \in \mathbb{N}$.

Elliptic curve crypto systems requires to choose one private key and generate the public key. Given base point P is multiplied by a scalar n and another point Q is calculated. In this scenario, n is called private and Q is called public key.

Definition 2.2.1. Determining n where $Q = nP$ is called elliptic curve discrete logarithm problem (Smart, 1999).

The security of elliptic curve cryptography relies on intractability of *elliptic curve discrete logarithm problem* which is introduced by Miller (Miller, 1985) and Koblitz (Koblitz, 1987).

To determine n by knowing Q and P is assumed to be intractable. The method for calculating n is to try every possible n value in the range $0 < n < \#E(\mathbb{K})$ ¹. The methods to find n , such as Pollard's Rho (Pollard, 1978), Pohling-Hellman (Pohlig & Hellman, 1978), Baby-step/giant-step (Odlyzko, 1984) have exponential time complexity for commonly used elliptic curves.

For both setting the public key Q and the attack which tries to determine secret key n depends on the scalar multiplication.

Definition 2.2.2. The scalar multiplication of P by n is shown below

$$[n]P = \underbrace{P + P + \dots + P}_{n \text{ times}}.$$

This operation which secures the elliptic curve cryptography also dominates the execution time. Thus, there are several ways to compute scalar multiplication of a point in terms of time and memory efficiency. One of the efficient methods is called sliding window scalar multiplication which makes use of pre-computations. See (Hankerson et al., 2003) for other methods.

¹ $\#E(\mathbb{K})$ denotes the number of points on the given elliptic curve.

Algorithm 1: Elliptic curve sliding window scalar multiplication	
Input:	$n, k \in \mathbb{N} \setminus \{0\}$ such that $n = (n_{l_1}, \dots, n_0)_2$, $P \in E$ over \mathbb{K} and $\{[3]P, [5]P, \dots, [2^k - 1]P\}$
Output:	$[n]P$
1	$Q \leftarrow [0]P, \quad i \leftarrow l - i$
2	repeat
3	if $n_i = 0$ then
4	$Q \leftarrow [2]Q, \quad i \leftarrow i - 1$
5	end
6	else
7	$s \leftarrow \max(i - k + 1, 0)$
8	repeat
9	$s \leftarrow s + 1$
10	until $n_s \neq 0$;
11	for h <i>from</i> 1 <i>to</i> $i - s + 1$ do
12	$Q \leftarrow [2]Q$
13	end
14	$u \leftarrow (n_i, \dots, n_s)_2, \quad Q \leftarrow Q + [u]P, \quad i \leftarrow s - 1$
15	end
16	until $i < 0$;
17	return Q

2.3 ISOGENIES

In (Vélu, 1971), Vélu introduced formulas for defining isogenies between two elliptic curves. After that, various researches focused on isogenies in terms of cryptographic operations. In this section, computing isogenies is explained in two different ways. First one is Vélu's formulas which makes use of list of points that is a finite order subgroup of an elliptic curve defined over a finite field. The second one is Kohel's approach (Kohel, 1996) which makes use the kernel of an isogeny. Both methods define the rational maps of isogenies and find codomain curve. The notation is obtained from (Silverman, 2009), (Washington, 2008), (Vélu, 1971) and (Kohel, 1996).

Definition 2.3.1. Let E_1 and E_2 are elliptic curves defined over \mathbb{K} . The morphism $\phi: E_1 \rightarrow E_2$ is called to be an isogeny from E_1 to E_2 that satisfies

$$\phi(\mathcal{O}) = \mathcal{O}.$$

If there is an isogeny from E_1 to E_2 with $\phi(E_1) \neq \{\mathcal{O}\}$ then E_1 and E_2 are said to be isogenous.

Definition 2.3.2. Let $\phi: E_1 \rightarrow E_2$ be an isogeny, then there exists its dual $\hat{\phi}: E_2 \rightarrow E_1$ such that $\hat{\phi} \circ \phi$ gives the map of multiplication by $\deg \phi$ on E_1 .

2.3.1 VÉLU'S FORMULAS

The domain and the co-domain of an isogeny can be determined by Vélu's formulas as follows.

Let \mathbb{K} be a field with a different characteristic than 2 and E_1 and E_2 be elliptic curves defined over \mathbb{K} .

$$E_1: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$.

Let ϕ be a separable isogeny from E_1 to E_2 and $C = \text{Ker } \phi$. Define $Q = (x_Q, y_Q) \in E_1$ with $Q \neq \infty$.

$$g_Q^x = 3x^2 + 2a_2x_Q + a_4 - a_1y_Q$$

$$g_Q^y = -2y_Q - a_1x_Q - a_3$$

$$vQ = g_Q^x \quad \text{where } 2Q = \infty \quad \text{or,}$$

$$vQ = 2g_Q^x - a_1g_Q^y \quad \text{where } 2Q \neq \infty$$

$$uQ = (g_Q^y)^2$$

Suppose that C_2 is the set of points of order 2 and R is the set of remaining

points in C . Now, calculate v and w as follows:

$$v = \sum_{Q \in S} v_Q, \quad w = \sum_{Q \in S} u_Q + x_Q v_Q$$

where $S = R \cup C_2$.

The isogeneous curve E_2 is defined as below

$$Y^2 + A_1XY + A_3Y = X^3 + A_2X^2 + A_4X + A_6$$

where $A_1 = a_1$, $A_2 = a_2$, $A_3 = a_3$, $A_4 = a_4 - 5v$, $A_6 = a_6 - (a_1^2 + 4a_2)v - 7w$.

The isogeny is given with the map $\phi: E_1 \rightarrow E_2$, $(x, y) \mapsto (X, Y)$ where

$$X = x + \sum_{Q \in S} \left(\frac{v_Q}{x - x_Q} + \frac{u_Q}{(x - x_Q)^2} \right)$$

$$Y = y - \sum_{Q \in S} \left(u_Q \frac{2y + a_1x + a_3}{(x - x_Q)^2} + v_Q \frac{a_1(x - x_Q + y - y_Q)}{(x - x_Q)^2} + \frac{a_1 + u_Q - g_Q^x g_Q^y}{(x - x_Q)^2} \right).$$

2.3.2 KOHEL'S APPROACH

The approach introduced by Kohel simplifies the way of calculating isogenies. Kohel showed that isogenies can be computed with the explicit functions in terms of the polynomial $\psi(X)$ which defines the ideal sheaf of the subgroup G of curve E defined over \mathbb{K} .

In this subsection, defining an explicit 2-isogeny from the domain curve $E_1: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ defined over \mathbb{K} is provided. Other scenarios can be found in (Kohel, 1996).

Let G be a subgroup of E_1 and defined by $\psi_1(x)$ that contains elements of order two. The isogeny of the subgroup H of degree 2 defined by $\psi_2(x) = \gcd(\psi_1(x), 4x^3 + b_2x^2 + 2b_4x + b_6)$ is determined as follows:

- $\psi_2(x) = x - x_0$ is a 2-isogeny from E_1 to a curve E_2 with the map $(x, y) \mapsto$

(X, Y) where

$$\begin{aligned} X &= x + \frac{3x_0^2 + 2a_2x_0 + a_4 - a_1y_0}{x - x_0} \\ Y &= y - (3x_0^2 + 2a_2x_0 + a_4 - a_1y_0) \frac{a_1(x - x_0) + (y - y_0)}{(x - x_0)^2}. \end{aligned} \quad (2.4)$$

y_0 defined with $y_0 = -\frac{a_1x_0 + a_3}{2}$ for characteristic different than 2.

- Let $t = 3x_0^2 + 2a_2x_0 + a_4 - a_1y_0$ and $w = x_0t$, then the codomain curve E_2 is written in the form

$$Y^2 + A_1XY + A_3Y = X^3 + A_2X^2 + A_4X + A_6,$$

where $A_1 = a_1$, $A_2 = a_2$, $A_3 = a_3$, $A_4 = a_4 - 5t$, $A_6 = a_6 - (a_1^2 + 4a_2)t - 7w$.



CHAPTER 3

EXTENDED HUFF CURVE

After Huff curves were introduced by Huff in 1948, this curve model has started to be used by crypto community with the work of Joye, Tibouchi and Vergnaud (Joye et al., 2010) at ANTS-IX. In that work, several interesting features such as *a more general curve model than Huff's original model, the explicit derivation of fast group operations, formulas for pairing computation and, an extension to the even characteristic case* were investigated. Wu and Feng (Wu & Feng, 2012) further extended the coverage to all elliptic curves having three points of order two.

Despite all the developments, any statement that indicates Huff curve form being faster than the other widely known forms of elliptic curves, does not exist in the relevant literature. And also, there is no consensus on how to write the curve equation for Huff form.

In this chapter, the Huff curve equation which is defined in a similar way with the other curve forms is presented. Subsequently, affine coordinates and projective closure of this form are given.

3.1 DERIVATION OF EXTENDED HUFF CURVE

Previous forms of Huff curves:

- The original Huff form was introduced as $ax(y^2 - 1) = by(x^2 - 1)$ by Huff in (Huff, 1948).
- A twisted version to cover more elliptic curves was given as $ax(y^2 - d) = by(x^2 - d)$ by Joye, Tibouchi and Vergnaud in (Joye et al., 2010).

- In a later work, the curve equation that covers even more elliptic curves was given as $x(ay^2 - 1) = y(bx^2 - 1)$ by Wu and Feng in (Wu & Feng, 2012)
- Finally, the curve equation that covers equally many elliptic curves as Wu and Feng's equation was given in the form $ax(y^2 - c) = by(x^2 - d)$ by Ciss and Sow in (Ciss & Sow, 2011)

Due to its extended coverage, Wu and Feng's equation is a good start. However, widely known curves like $ax^2 + y^2 = 1 + dx^2y^2$ (twisted Edwards form) (Bernstein et al., 2008), $y^2 = dx^4 + 2ax^2 + 1$ (extended Jacobi quartic form) (Billet & Joye, 2003), $ax^3 + y^3 + 1 = dxy$ (twisted Hessian form) (Bernstein et al., 2015), $y^2 = x^3 + ax + b$ (short Weierstrass form), etc. are always written with “+” in their curve equations in the literature. In order to make Huff curve to compile better with literature, by making a tweak which does not affect the coverage, the equation can be written as $x(1 + ay^2) = y(1 + bx^2)$.

In all speed oriented formulas, both a and b do appear with respect to the equation $x(1 + ay^2) = y(1 + bx^2)$. So, keeping a and b as small as possible is preferable. But, this case cause the number of elliptic curves over some suitable field to be very limited. In Ciss and Sow's equation, the constants appear outside the parentheses. This is very helpful in terms of increase the coverage. But, both constants clash with a and b of Wu and Feng's equation. Thus, in order to prevent the clashing of constants, the curve can be written as $cx(1 + ay^2) = dy(1 + bx^2)$. Since a point satisfying this equation also satisfies $(c/d)x(1 + ay^2) = y(1 + bx^2)$, having just c is enough. So, one can rename c/d as c and obtain $cx(1 + ay^2) = y(1 + bx^2)$.

The last phase is to keep a as is and replace b with d , because the letter a and d are more common to use in curve forms. The final version of the curve form is $y(1 + ax^2) = cx(1 + dy^2)$. In order to prevent ambiguation, this equation is referred as extended Huff curve where the curve constants appear in alphabetic order. This form has the same coverage as Wu and Feng's equation, i.e. extended Huff form covers all elliptic curves having three points of order two.

3.2 AFFINE COORDINATES OF EXTENDED HUFF CURVE

Definition 3.2.1. Let \mathbb{K} be a field with a characteristic different than 2 and $a, c, d \in \mathbb{K}$. An extended Huff curve over \mathbb{K} is defined as follows:

$$H_{a,c,d}: y(1 + ax^2) = cx(1 + dy^2). \quad (3.1)$$

- $H_{a,c,d}$ is non-singular if and only if the condition $acd(a - c^2d) \neq 0$ is satisfied.
- The j -invariant of $H_{a,c,d}$ is $256(a^2 - ac^2d + c^4d^2)^3 / (ac^2d(a - c^2d))^2$.

Theorem 3.2.1. Every elliptic curve having three points of order two is isomorphic over \mathbb{K} to an extended Huff curve.

Proof. Let $e, f \in \mathbb{K}$ and $e' = a - c^2d$ and $f' = -c^2d$. An extended Huff curve $H_{a,c,d}$ with identity element at $(0, 0)$ is isomorphic over \mathbb{K} to a Weierstrass curve of the form

$$W_{e,f}: y^2 = x(x - e)(x - f)$$

with identity element at $(0 : 1 : 0)$. The isomorphism maps are given below.

$$\begin{aligned} \phi: H \rightarrow W, \quad (x, y) &\mapsto \left(cx \frac{a - c^2d}{cx - y}, c \frac{a - c^2d}{cx - y} \right), \\ \phi^{-1}: W \rightarrow H, \quad (x, y) &\mapsto \left(\frac{x}{y}, c \frac{x - (a - c^2d)}{y} \right). \end{aligned} \quad (3.2)$$

Since every elliptic curve having three points of order two is isomorphic over \mathbb{K} to $W_{e,f}$, those curves are also isomorphic over \mathbb{K} to $H_{a,c,d}$. \square

Note that, the curve constants a, d can be scaled to small field elements. Let $\alpha, \delta, a', d' \in \mathbb{K}$ and a, d are small elements of the field \mathbb{K} , which satisfies $a = \alpha^2 a'$ and $d = \delta^2 d'$. The isomorphism map of the rescaling is as follows:

$$H_{a,c,d} \rightarrow H_{a', \frac{\delta}{\alpha}c, d'}, \quad (x, y) \mapsto (\alpha x, \delta y). \quad (3.3)$$

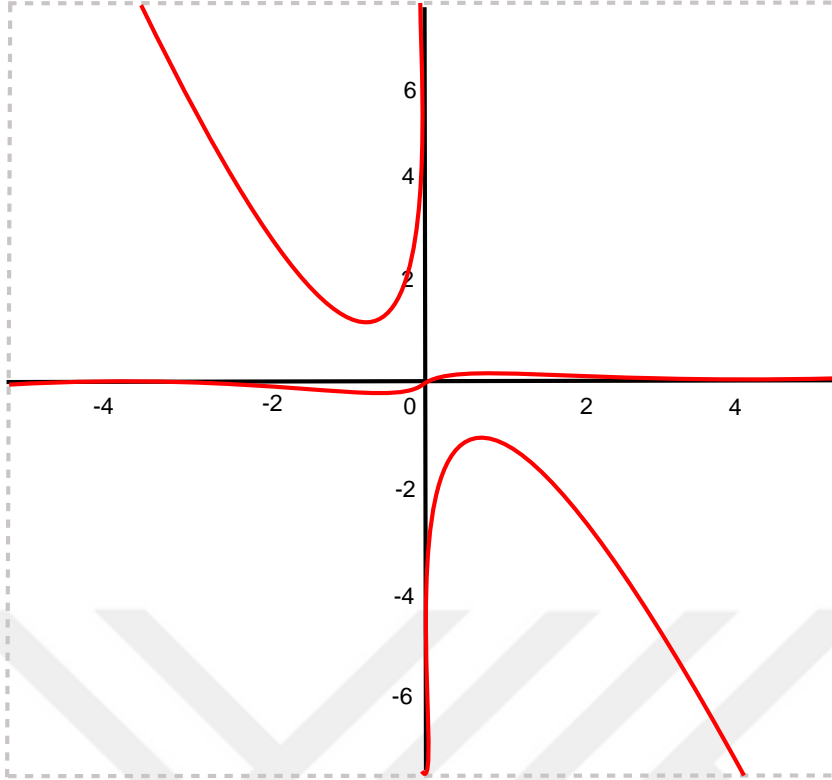


Figure 3.1: Huff curve $y(1 + 23x^2) = 10x(1 + 25y^2)$ over \mathbb{R} .

Furthermore, the curve constant c can be scaled to 1 by substituting x/c to x . In this case the isomorphism maps in Theorem 3.2.1 take a simpler form. But, keeping c as it is and scaling a, d to small elements is more advantageous in terms of faster field operations.

3.2.1 GROUP LAW IN AFFINE COORDINATES OF EXTENDED HUFF CURVE

A chord and tangent rule can be defined on an extended Huff curve defined over field \mathbb{K} with $\text{char} \neq 2$. This rule is for adding two points on the curve to obtain the third point (sum of initial two points) on the curve. This set of points constitutes an abelian group.

The addition of two distinct points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on extended Huff curve can be visualized easily by the use of chord and tangent rule. A line that intersects the points P and Q , must intersect the curve at a third point by Bézout's Theorem (Bézout, 1779). The reflection of this point about $(0, 0, 1)$

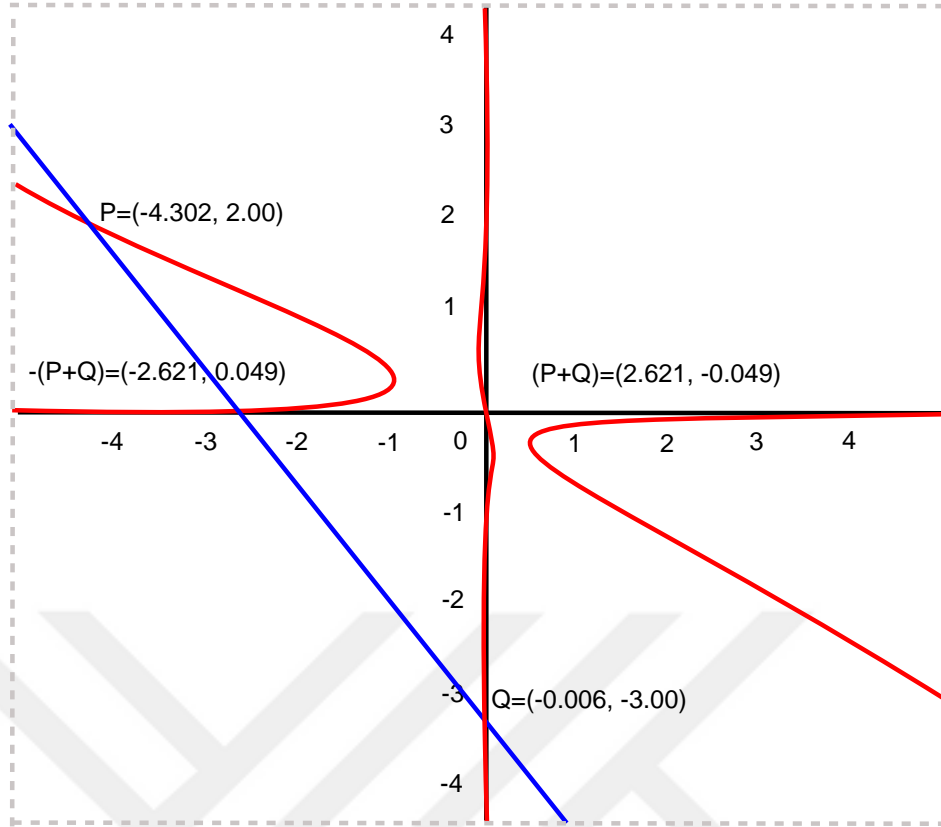


Figure 3.2: Chord and Tangent Rule on extended Huff curve $y(1 + 24x^2) = -3x(1 + 17y^2)$ over \mathbb{R} .

As shown in figure, A line passes through point P and Q intersects the curve at a third point $-(P + Q)$. The reflection of this intersection point about $(0, 0, 1)$ gives the sum of P and Q . This rule is verified by the Formulas 3.4.

gives $P + Q$.

This rule is also applicable for doubling. The tangent line of the point P intersects the curve at a second point. The reflection of this point about $(0, 0, 1)$ gives $[2]P$.

- **Identity element:**

The identity element of $H_{a,c,d}$ is $\mathcal{O} = (0 : 1 : 0)$.

- **Points at infinity:**

There are three points at infinity of the curve $H_{a,c,d}$ such that $(0 : 1 : 0)$, $(1 : 0 : 0)$ and $(cd : a : 0)$. These points respectively correspond to the three points of order two $(0, 0)$, $(e', 0)$, $(f', 0)$ on $W_{e,f}$.

- **Negative of a point:**

The negative of a point $P = (x, y)$ on $H_{a,c,d}$ is shown as $-P = (-x, -y)$.

- **Addition on $H_{a,c,d}$** : Two different addition formulas are given for $H_{a,c,d}$ such that dedicated addition and unified addition. While unified addition formulas can be used for addition of the identical points on curve, dedicated addition can only be used for distinct points.

1. **Dedicated addition** :

Let the points $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ be on $H_{a,c,d}$ and $P_1 \neq P_2$.

The affine addition formulas are represented as $P_1 + P_2 = P_3 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= \frac{(x_1 - x_2)(y_1 + y_2)}{(1 + ax_1x_2)(y_1 - y_2)}, \\ y_3 &= \frac{(x_1 + x_2)(y_1 - y_2)}{(x_1 - x_2)(1 + dy_1y_2)}. \end{aligned} \quad (3.4)$$

2. **Unified addition** : The alternative addition formulas of the points

P_1 and P_2 are represented as $P_3 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= \frac{(x_1 + x_2)(1 - dy_1y_2)}{(1 - ax_1x_2)(1 + dy_1y_2)}, \\ y_3 &= \frac{(1 - ax_1x_2)(y_1 + y_2)}{(1 + ax_1x_2)(1 - dy_1y_2)}. \end{aligned} \quad (3.5)$$

It is assumed that $ax_1x_2 \neq \pm 1$ and $dy_1y_2 \neq \pm 1$. Note that this set of formulas can be used for identical points and as well as doubling.

- **Doubling on $H_{a,c,d}$** :

The doubling formulas of the point $P_1 = (x_1, y_1)$ is represented as $[2](x_1, y_1) = (x_3, y_3)$ where

$$\begin{aligned} x_3 &= \frac{2x_1(1 - dy_1^2)}{(1 - ax_1^2)(1 + dy_1^2)}, \\ y_3 &= \frac{2y_1(1 - ax_1^2)}{(1 + ax_1^2)(1 - dy_1^2)}. \end{aligned} \quad (3.6)$$

The doubling can be calculated with the assumptions $ax_1^2 \neq \pm 1$ and $dy_1^2 \neq$

± 1 .

3.3 PROJECTIVE COORDINATES OF EXTENDED HUFF CURVE

This section comprises of both \mathbb{P}^2 and $\mathbb{P}^1 \times \mathbb{P}^1$ embeddings of extended Huff curve. To embed Huff curve into $\mathbb{P}^1 \times \mathbb{P}^1$ is a quite new concept and it provides the performance improvement. Unified addition and doubling formulas are of lower total degree and 4-way parallel by nature when $\mathbb{P}^1 \times \mathbb{P}^1$ embedding is used rather than \mathbb{P}^2 . In this section, only the formulas are given. Efficient implementations of each formula can be found in Chapter 5.

Note that, the formulas for \mathbb{P}^2 are derived from (Joye et al., 2010), the only difference is the curve constants that are located in doubling and addition formulas. The reason is the different sequence of curve constants in curve equation, this situation explained detailed at the beginning of the chapter.

3.3.1 EMBEDDING H INTO \mathbb{P}^2

The projective closure of $H_{a,c,d}$ in \mathbb{P}^2 is defined as follows:

$$\hat{H} = \{(X : Y : Z) \in \mathbb{P}^2 : Y(aX^2 + Z^2) - cX(dY^2 + Z^2)\}. \quad (3.7)$$

Let $\lambda \in \mathbb{K}$ and $\lambda \neq 0$. In the \mathbb{P}^2 embedding of the curve $H_{a,c,d}$, a point (x, y) is represented with $(\lambda x : \lambda y : \lambda)$. This point corresponds to the affine point $((\lambda x/\lambda), (\lambda y/\lambda))$.

Group Law in \mathbb{P}^2 embedding of Extended Huff Curves

- **Identity element:**

The identity element of \hat{H} is $\hat{H}_{\mathcal{O}} = (0 : 1 : 0)$.

- **Points at infinity:**

There are three points at infinity of the curve \hat{H} such that $(0: 1: 0)$, $(1: 0: 0)$ and $(cd: a: 0)$.

- **Negative of a point:**

The negative of a point $P = (X: Y: Z)$ on \mathcal{H} is shown as $-P = (-X: -Y: Z)$.

- **Addition on \hat{H} :** The projective forms of the addition formulas given in Formula 3.4 and Formula 3.5 are given in the following part.

1. **Dedicated addition :**

Let the points $P_1 = (X_1: Y_1: Z_1)$, $P_2 = (X_2: Y_2: Z_2)$ be on \hat{H} and $P_1 \neq P_2$. The dedicated addition formulas are represented as $P_1 + P_2 = P_3 = (X_3: Y_3: Z_3)$:

$$\begin{aligned} X_3 &= (X_1Z_2 - X_2Z_1)^2(X_1Z_2 + Y_2Z_1)(Z_1Z_2 + dX_1Y_2) \\ Y_3 &= (Y_1Z_2 - Y_2Z_1)^2(X_1Z_2 + X_2Z_1)(Z_1Z_2 + aX_1X_2) \\ Z_3 &= (Y_1Z_2 - Y_2Z_1)(Z_1Z_2 + aX_1X_2)(X_1Z_2 - X_2Z_1)(Z_1Z_2 + dX_1Y_2). \end{aligned} \tag{3.8}$$

2. **Unified addition :** The alternative addition formulas of the points

P_1 and P_2 are represented as $P_3 = (X_3: Y_3: Z_3)$:

$$\begin{aligned} X_3 &= (X_1Z_2 + X_2Z_1)(dY_1Y_2 - Z_1Z_2)^2(Z_1Z_2 + aX_1X_2) \\ Y_3 &= (Y_1Z_2 + Y_2Z_1)(aX_1X_2 - Z_1Z_2)^2(Z_1Z_2 + dY_1Y_2) \\ Z_3 &= (aX_1X_2 - Z_1Z_2)(Z_1Z_2 + dY_1Y_2)(Z_1Z_2 + aX_1X_2)(dY_1Y_2 - Z_1Z_2). \end{aligned} \tag{3.9}$$

Note that this set of formulas can be used for identical points and as well as doubling.

- **Doubling on \hat{H} :**

The doubling formulas of the point $P_1 = (X_1: Y_1: Z_1)$ is represented as

$[2]P_1 = P_1 = (X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ where

$$\begin{aligned} X_3 &= 2X_1(dY_1^2 - Z_1^2)^2 Z_1(Z_1^2 + aX_1^2) \\ Y_3 &= 2Y_1(aX_1^2 - Z_1^2)^2 Z_1(Z_1^2 + dY_1^2) \\ Z_3 &= (aX_1^2 - Z_1^2)(Z_1^2 + dY_1^2)(Z_1^2 + aX_1^2)(dY_1^2 - Z_1^2). \end{aligned} \quad (3.10)$$

3.3.2 EMBEDDING H INTO $\mathbb{P}^1 \times \mathbb{P}^1$

The projective closure of $H_{a,c,d}$ in $\mathbb{P}^1 \times \mathbb{P}^1$ is defined as follows:

$$\mathcal{H} = \{((X : Z), (Y : T)) \in \mathbb{P}^1 \times \mathbb{P}^1 : YT(Z^2 + aX^2) = cXZ(T^2 + dY^2)\}. \quad (3.11)$$

Let $\lambda, \delta \in \mathbb{K}$ and $\lambda, \delta \neq 0$. In the $\mathbb{P}^1 \times \mathbb{P}^1$ embedding of the curve $H_{a,c,d}$, a point (x, y) is represented with $((\lambda x : \lambda), (\delta y : \delta))$. This point corresponds to the affine point $((\lambda x / \lambda), (\delta y / \delta))$.

Group Law in $\mathbb{P}^1 \times \mathbb{P}^1$ embedding of Extended Huff Curves

- **Identity element:**

The identity element of \mathcal{H} is $\mathcal{H}_O = ((0 : 1), (0 : 1))$.

- **Points at infinity:**

There are three points at infinity of the curve \mathcal{H} such that $((0 : 1), (1 : 0)), ((1 : 0), (0 : 1))$ and $((1 : 0), (1 : 0))$. These points respectively correspond to the three points of order two $(0, 0), (e', 0), (f', 0)$ on $W_{e,f}$ given in the proof of Theorem 3.2.1.

- **Negative of a point:**

The negative of a point $P = ((X : Z), (Y : T))$ on \mathcal{H} is shown as $-P = ((-X : Z), (-Y : T))$.

- **Addition on \mathcal{H} :** The projective forms of the addition formulas given in Formula 3.4 and Formula 3.5 are given in the following part.

1. **Dedicated addition :**

Let the points $P_1 = ((X_1: Z_1), (Y_1: T_1))$, $P_2 = ((X_2: Z_2), (Y_2: T_2))$ be on \mathcal{H} and $P_1 \neq P_2$. The dedicated addition formulas are represented as $P_1 + P_2 = P_3 = ((X_3: Z_3), (Y_3: T_3))$:

$$\begin{aligned} X_3 &= (X_1Z_2 - Z_1X_2)(Y_1T_2 + T_1Y_2) \\ Z_3 &= (Z_1Z_2 + aX_1X_2)(Y_1T_2 - T_1Y_2) \\ Y_3 &= (X_1Z_2 + Z_1X_2)(Y_1T_2 - T_1Y_2) \\ T_3 &= (X_1Z_2 - Z_1X_2)(T_1T_2 + dY_1Y_2). \end{aligned} \quad (3.12)$$

2. **Unified addition :** The alternative addition formulas of the points P_1 and P_2 are represented as $P_3 = ((X_3: Z_3), (Y_3: T_3))$:

$$\begin{aligned} X_3 &= (X_1Z_2 + Z_1X_2)(T_1T_2 - dY_1Y_2) \\ Z_3 &= (Z_1Z_2 - aX_1X_2)(T_1T_2 + dY_1Y_2) \\ Y_3 &= (Z_1Z_2 - aX_1X_2)(Y_1T_2 + T_1Y_2) \\ T_3 &= (Z_1Z_2 + aX_1X_2)(T_1T_2 - dY_1Y_2). \end{aligned} \quad (3.13)$$

Note that this set of formulas can be used for identical points and as well as doubling.

• **Doubling on \mathcal{H} :**

The doubling formulas of the point $P_1 = ((X_1: Z_1), (Y_1: T_1))$ is represented as $[2]P_1 = ((X_1: Z_1), (Y_1: T_1)) = ((X_3: Z_3), (Y_3: T_3))$ where

$$\begin{aligned} X_3 &= 2X_1Z_1(T_1^2 - dY_1^2) \\ Z_3 &= (Z_1^2 - aX_1^2)(T_1^2 + dY_1^2) \\ Y_3 &= 2Y_1T_1(Z_1^2 - aX_1^2) \\ T_3 &= (Z_1^2 + aX_1^2)(T_1^2 - dY_1^2). \end{aligned} \quad (3.14)$$

CHAPTER 4

2-ISOGENY ON EXTENDED HUFF CURVE

Elliptic curve isogenies are very useful in terms of cryptographic implementations. The cryptographic operations such as point counting, analyzing the difficulty of elliptic curve discrete logarithm problem, random number generation and hash functions use isogenies as a tool. One of the most significant feature of elliptic curve isogenies is that gives the multiplication by n -map which is defined in 2.3.2. Thus, isogenies are used to optimize doubling map of extended Huff curve in this work. In the first section, the adoption of Kohel's approach is explained. And then, in the second section a 2-isogeny is defined from an extended Huff curve to another extended Huff curve. The obtained map provides the fastest doubling map on the curve. In the last section, a 2-isogeny is defined from an extended Huff curve to twisted Edwards curve. This, gives a doubling map which is almost fast as the previous map, and also gives an alternative doubling map for twisted Edwards curve.

4.1 DEFINING ISOGENY WITH KOHEL'S APPROACH

Kohel's approach, explained in 2.3.2, shows how to define isogenies between two Weierstrass form of elliptic curves. An extended Huff curve $H_{a,c,d}$ is isomorphic over \mathbb{K} to a Weierstrass curve of the form $W_{e,f}: y^2 = x(x - e)(x - f) = x^3 - (e + f)x^2 + (ef)x$ with the map given in 3.2. Thus, a 2-isogeny can be easily defined from an extended Huff curve to a Weierstrass curve by making use of this isomorphism and Kohel's approach.

Let \mathbb{K} is a field with a characteristic different than 2 and $a_1, a_2, a_3, a_4, a_6, e, f \in \mathbb{K}$. Weierstrass curves $E_W: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ and $W_{e,f}: y^2 =$

$x^3 - (e + f)x^2 + (ef)x$ are defined over \mathbb{K} .

- The coefficients of E_W can be written as follows:

$$a_1 = 0, \quad a_2 = -(e + f), \quad a_3 = 0, \quad a_4 = ef, \quad a_6 = 0.$$

- Point $Q = (x_0, y_0) = (0, 0) \in \psi(x)$ which is the subgroup of E_W that contains points of order two.
- In this case, the coefficients of the isogenous curve are written as:

$$A_1 = 0, \quad A_2 = -(e + f), \quad A_3 = 0, \quad A_4 = -4ef, \quad A_6 = 4ef(e + f).$$

- So, the isogenous curve equation is

$$W': y^2 = x^3 - (e + f)x^2 - 4efx - 4ef(e + f).$$

- Point Q corresponds to the point $Q' = (X, Y)$ on the codomain curve W' is expressed as

$$\begin{aligned} X &= x + \frac{ef}{x}, \\ Y &= y - \frac{efy}{x^2} \end{aligned}$$

which is computed with the map given in 2.4.

These steps lead to write the isogeny as follows:

The curve $W: y^2 = x^3 - (e + f)x^2 + (ef)x$ is 2-isogenic to the curve $W': y^2 = x^3 - (e + f)x^2 - 4efx - 4ef(e + f)$ with the map

$$\Gamma: W \rightarrow W', (x, y) \mapsto \left(x + \frac{ef}{x}, y - \frac{efy}{x^2} \right).$$

Its dual can be calculated with the same manner.

As it is mentioned at the beginning of this section, the domain curve is isomorphic to the extended Huff curve. It means that an isogeny can be defined from extended Huff curve to the codomain curve W' with the composition of isomorphism and the isogeny maps defined above. Therefore, 2-isogenies can be defined from extended Huff curve to any curve form which has three points of order two. It can be obtained by the composition of the isomorphism maps given in Chapter 2 and the isogeny map defined with Kohel's approach. In the following sections it will be explained in detail.

4.2 ISOGENY TO EXTENDED HUFF CURVE

In (Moody & Shumow, 2001), it is shown that a 2-isogeny can be derived from a Huff curve to another Huff curve. It is realized that more speed oriented isogenies can be derived. The detailed instructions to define a 2-isogeny from an extended Huff curve to another extended Huff curve are given in this section.

4.2.1 CONSTRUCTING ISOGENY MAPS BETWEEN 2 DIFFERENT EXTENDED HUFF CURVES

A degree 2 isogeny can be defined from a Weierstrass curve of the form $W: y^2 = x(x - e)(x - f)$ to another Weierstrass curve of the form $\hat{W}: y^2 = x(x - \hat{e})(x - \hat{f})$. Also, by Theorem 3.2.1, it is known that every extended Huff curve H is isomorphic over \mathbb{K} to W . Thus, the below substitution can be done between the curve constants of \hat{W} and H .

Let $r \in \mathbb{K}$, $r^2 = ad$, $\hat{e} = -(a - cr)^2/a$ and $\hat{f} = -(a + cr)^2/a$. The isogeny from W to \hat{W} and its dual are given below:

$$\begin{aligned}\mu: W &\rightarrow \hat{W}, & (x, y) &\mapsto \left(\frac{x^2 - \hat{e}x}{x - \hat{f}}, y + \frac{\hat{f}(\hat{e} - \hat{f})}{(x - \hat{f})^2}y \right) \\ \hat{\mu}: \hat{W} &\rightarrow W, & (x, y) &\mapsto \left(\frac{(x + \hat{e})^2}{4x}, \frac{x^2 - \hat{e}^2}{8x^2}y \right).\end{aligned}$$

The combination of the isomorphism map ϕ from the Theorem 3.2.1 and μ gives an isogeny from H to \hat{W} . And the combination of ϕ^{-1} and $\hat{\mu}$ gives its dual. Some birational transformations help to put \hat{W} in desired extended Huff curve form.

First step is to find an isomorphic Legendre form $\tilde{W}: y^2 = x(x - \tilde{e})(x - \tilde{f})$ to \hat{W} . By moving the point $(\tilde{f}, 0)$ on \hat{W} to $(\tilde{e}, 0)$ leads to obtain the maps between these two isomorphic curves.

$$\begin{aligned}\psi: \hat{W} &\rightarrow \tilde{W}, & (x, y) &\mapsto (x + \tilde{f}, y), \\ \psi^{-1}: \tilde{W} &\rightarrow \hat{W}, & (x, y) &\mapsto (x - \tilde{f}, y)\end{aligned}$$

where $\tilde{e} = -4cr$ and $\tilde{f} = (a - cr)^2/a$.

Since the aim is to reach another extended Huff curve form, a reverse manner of Theorem 3.2.1 is used. Let $\tilde{a} = -(a + cr)^2/a$ and $\tilde{d} = -(a - cr)^2/a$. Then the extended Huff curve of the form $\tilde{H}: y(1 + \tilde{a}x^2) = x(1 + \tilde{d}y^2)$ is isomorphic to \tilde{W} . The isomorphism map and its inverse is as follows:

$$\begin{aligned}\tau: \tilde{W} &\rightarrow \tilde{H}, & (x, y) &\mapsto \left(\frac{x}{y}, \frac{x - (\tilde{a} - \tilde{d})}{y} \right), \\ \tau^{-1}: \tilde{H} &\rightarrow \tilde{W}, & (x, y) &\mapsto \left(\frac{(\tilde{a} - \tilde{d})x}{x - y}, \frac{\tilde{a} - \tilde{d}}{x - y} \right).\end{aligned}$$

Final step is to rescale the constants in order to obtain isogenic extended Huff curve

$$G: y(1 - ax^2) = \left(\frac{a - cr}{a + cr} \right) x(1 - ay^2)$$

by using the maps below:

$$\begin{aligned}\omega: \tilde{H} &\rightarrow G, & (x, y) &\mapsto \left(\frac{(a+cr)x}{a}, \frac{(a-cr)y}{a} \right), \\ \omega^{-1}: G &\rightarrow \tilde{H}, & (x, y) &\mapsto \left(\frac{ax}{a+cr}, \frac{ay}{a-cr} \right).\end{aligned}$$

Hence,

- the domain curve is $H: y(1+ax^2) = cx(1+dy^2)$
- the codomain curve is $G: y(1-ax^2) = \left(\frac{a-cr}{a+cr} \right) x(1-ay^2)$.

The 2-isogeny map φ' from H to G and its dual $\hat{\varphi}'$ can be derived by

$$\begin{aligned}\varphi' &= (\omega \circ \tau \circ \psi \circ \mu \circ \phi), \\ \hat{\varphi}' &= (\phi^{-1} \circ \hat{\mu} \circ \psi^{-1} \circ \tau^{-1} \circ \omega^{-1}).\end{aligned}$$

The following theorem proves the isogeny map given above.

Theorem 4.2.1. Let $a, c, d, r \in \mathbb{K}$, $r^2 = ad$ and $acd(a-c^2d) \neq 0$ Then, the curve

$$H: y(1+ax^2) = cx(1+dy^2)$$

is 2-isogenous over \mathbb{K} to the extended Huff curve

$$G: y(1-ax^2) = \left(\frac{a-cr}{a+cr} \right) x(1-ay^2).$$

The 2-isogeny and its dual are given explicitly as follows:

$$\begin{aligned}\varphi: H &\rightarrow G, & (x, y) &\mapsto \left(\frac{x + \frac{r}{a}y}{1+rxy}, \frac{x - \frac{r}{a}y}{1-rxy} \right), \\ \hat{\varphi}: G &\rightarrow H, & (x, y) &\mapsto \left(\frac{x+y}{1-axy}, \frac{x-y}{1+axy} \cdot \frac{a}{r} \right).\end{aligned}$$

Proof.

- In order G to be an extended Huff curve, first thing to show is that the $\Delta_G \neq 0$.

$$- \Delta_H = acd(a - c^2d) \neq 0 \quad \rightarrow \quad a - c^2d \neq 0 \text{ and } c \neq 0.$$

$$- (a/c)^2 \neq ad = r^2 \quad \rightarrow \quad r \neq \pm a/c.$$

$$- a \pm cr \neq 0 \quad \rightarrow \quad (a - cr)/(a + cr) \neq 0.$$

$$- r \neq 0 \text{ and } a + cr \neq 0 \quad \rightarrow \quad (-a)((-a) - ((a - cr)/(a + cr))^2(-a)) = 2acr/(a + cr) \neq 0$$

So that, Δ_G is obtained as nonzero.

$$\Delta_G = (-a) \left(\frac{a - cr}{a + cr} \right) (-a) \left((-a) - \left(\frac{a - cr}{a + cr} \right)^2 (-a) \right) \neq 0.$$

- Secondly, show that φ is an isogeny from H to G .

1. By substituting u with $\frac{x + \frac{r}{a}y}{1 + rxy}$ and v with $\frac{x - \frac{r}{a}y}{1 - rxy}$, it is obtained that

$$v(1 - au^2) - \left(\frac{a - cr}{a + cr} \right) u(1 - av^2) = \frac{-2r(1 - ax^2)(a - r^2y^2)(ay(1 + ax^2) - cx(a + r^2y^2))}{a^2(a + cr)(1 - r^2x^2y^2)^2}.$$

2. By substituting r^2 with ad and organizing the terms, it is obtained that

$$\left(y(1 + ax^2) - cx(1 + dy^2) \right) \cdot \frac{-2r(1 - ax^2)(1 - dy^2)}{(a + cr)(1 - adx^2y^2)^2}.$$

It means that, φ is a rational map from H to G .

3. $\varphi((0, 0)) = (0, 0)$ shows that φ is an isogeny from H to G .

- Next step is to show that $\hat{\varphi}$ is an isogeny from G to H .

1. By substituting u with $\frac{x + y}{1 - axy}$ and v with $\frac{a}{r} \cdot \frac{x - y}{1 + axy}$, it is obtained

that $v(1 + au^2) - u(1 + dv^2) =$

$$\begin{aligned} & -\left(c(x+y)(1-axy)(1+axy)^2/(r-ra^2x^2y^2)^2\right)r^2+ \\ & \left(a(x-y)(1+axy)(1+ax^2)(1+ay^2)/(r-ra^2x^2y^2)^2\right)r- \\ & \left(a^2cd(x+y)(1-axy)(x-y)^2/(r-ra^2x^2y^2)^2\right). \end{aligned}$$

2. By substituting d with r^2/a and organizing the terms, it is obtained that

$$\left(y(1-ax^2) - \left(\frac{a-cr}{a+cr}\right)x(1-ay^2)\right) \cdot \frac{(a+cr)(1+ax^2)(1+ay^2)}{-r(1-a^2x^2y^2)^2}$$

It means that, $\hat{\varphi}$ is a rational map from G to H .

3. $\hat{\varphi}((0,0)) = (0,0)$ shows that $\hat{\varphi}$ is an isogeny from G to H .

- By Definition 2.3.2, it must be shown that composition of the isogeny map and its dual gives the multiplication map.

1.

$$\hat{\varphi} \circ \varphi = \left(\frac{-2xy(ax-cdy)(ax^2-2cdxy+dy^2)}{(ax^2-dy^2)(acx^2-2axy+cdy^2)}, \frac{-2xy(ax-cdy)(acx^2-2axy+cdy^2)}{c(ax^2-dy^2)(ax^2-2cdxy+dy^2)} \right).$$

By substituting c with

$$y(1+ax^2)/(x(1+dy^2))$$

in the map above, it can be obtained that $\hat{\varphi} \circ \varphi = [2]_H \in \mathbb{K}(H)$

2. With the same manner above, eliminate c by using the relation

$$\frac{ay(1-ax^2) - ax(1-ay^2)}{ry(1-ax^2) + rx(1-ay^2)}$$

and eliminate d by using the relation $d = r^2/a$ in $\varphi \circ \hat{\varphi}$. Then, the remaining expression gives $[2]_G \in \mathbb{K}(E)$.

□

4.2.2 2-ISOGENY IN $\mathbb{P}^1 \times \mathbb{P}^1$

The $\mathbb{P}^1 \times \mathbb{P}^1$ embedding of the curve G is

$$\mathcal{G}: YT(Z^2 - aX^2) = \left(\frac{1-c}{1+c}\right)XZ(T^2 - aY^2).$$

The projective Huff curve \mathcal{H} is 2-isogeneous over \mathbb{K} to \mathcal{G} . The 2-isogeny map and its dual are given as

$$\begin{aligned} \varphi_{\mathcal{H}}: \mathcal{H} &\rightarrow \mathcal{G}, & ((X: Z), (Y: T)) &\mapsto ((XT + YZ: TZ + aXY), (XT - YZ: TZ - aXY)), \\ \hat{\varphi}_{\mathcal{G}}: \mathcal{G} &\rightarrow \mathcal{H}, & ((X: Z), (Y: T)) &\mapsto ((XT + YZ: TZ - aXY), (XT - YZ: TZ + aXY)), \end{aligned} \tag{4.1}$$

Note that, in order to obtain faster arithmetic, a is chosen to be equal to d , and so $r = \pm a$. In Chapter 5, r is assumed to be equal to a .

4.3 ISOGENY TO TWISTED EDWARDS CURVE

The detailed instructions to define a 2-isogeny from an extended Huff curve to a twisted Edwards curve are given in this section.

4.3.1 CONSTRUCTING ISOGENY MAPS BETWEEN EXTENDED HUFF CURVE AND TWISTED EDWARDS CURVE

A degree 2 isogeny can be defined from a Weierstrass curve of the form $W: y^2 = x(x - e')(x - f')$ to another Weierstrass curve of the form $\hat{W}: y^2 = x^3 + 2(e' +$

$f')x^2 + (e' - f')^2x$. The isogeny from W to \hat{W} and its dual are given below:

$$\begin{aligned}\gamma: W &\rightarrow \hat{W}, & (x, y) &\mapsto \left(\frac{x^2 - (e' + f')x + e'f'}{x}, y \frac{x^2 - e'f'}{x^2} \right), \\ \hat{\gamma}: \hat{W} &\rightarrow W, & (x, y) &\mapsto \left(\frac{x^2 + 2(e' + f')x + (e' - f')^2}{4x}, y \frac{x^2 - (e' - f')^2}{8x^2} \right).\end{aligned}$$

The combination of the isomorphism map ϕ from the Theorem 3.2.1 and γ gives an isogeny from H to \hat{W} . And the combination of ϕ^{-1} and $\hat{\gamma}$ gives its dual. Some birational transformations help to put \hat{W} in desired twisted Edwards curve form.

First step is to find an isomorphic Montgomery form $M: (e' - f')y^2 = x^3 + 2\frac{(e'+f')}{(e'-f')}x^2 + x$ to \hat{W} .

$$\begin{aligned}\epsilon: \hat{W} &\rightarrow M, & (x, y) &\mapsto \left(\frac{x}{(e' - f')}, \frac{y}{(e' - f')^2} \right), \\ \epsilon^{-1}: M &\rightarrow \hat{W}, & (x, y) &\mapsto \left(\frac{x}{(e' - f')^{-1}}, \frac{y}{(e' - f')^{-2}} \right).\end{aligned}$$

Since the aim is to reach a twisted Edwards curve form, put M in twisted Edwards form

$$E': \quad \frac{4e'}{(e' - f')^2}x^2 + y^2 = 1 + \frac{4f'}{(e' - f')^2}x^2y^2$$

The isomorphism map and its inverse is as follows: (Bernstein et al., 2008)

$$\begin{aligned}\lambda: M &\rightarrow E', & (x, y) &\mapsto \left(\frac{x}{y}, \frac{x-1}{x+1} \right), \\ \lambda^{-1}: E' &\rightarrow M, & (x, y) &\mapsto \left(\frac{1+y}{1-y}, \frac{1+y}{x(1-y)} \right).\end{aligned}$$

Rescale the constants to get rid of $4/(e' - f')^2$,

$$\tilde{E}: \quad e'x^2 + y^2 = 1 + f'x^2y^2$$

using the maps from (Bernstein et al., 2008)

$$\begin{aligned}\xi: E' &\rightarrow \tilde{E}, & (x, y) &\mapsto \left(\frac{2}{e' - f'}x, y\right), \\ \xi^{-1}: \tilde{E} &\rightarrow E', & (x, y) &\mapsto \left(\frac{e' - f'}{2}x, y\right).\end{aligned}$$

$$G: y(1 - ax^2) = \left(\frac{a - cr}{a + cr}\right)x(1 - ay^2)$$

In order to obtain a common form of a twisted Edwards curve, the following steps are required. Swap e' and f' by

$$\hat{E}: f'x^2 + y^2 = 1 + e'x^2y^2$$

using the maps from (Bernstein et al., 2008)

$$\begin{aligned}\pi: \tilde{E} &\rightarrow \hat{E}, & (x, y) &\mapsto (x, y^{-1}), \\ \pi^{-1}: \hat{E} &\rightarrow \tilde{E}, & (x, y) &\mapsto (x, y^{-1}).\end{aligned}$$

Finally, in order to remove c^2 , set $e = -d$ and $f = a/c^2 - d$.

$$E: ex^2 + y^2 = 1 + fx^2y^2$$

using the maps from (Bernstein et al., 2008)

$$\begin{aligned}\eta: \hat{E} &\rightarrow E, & (x, y) &\mapsto (cx, y), \\ \eta^{-1}: E &\rightarrow \hat{E}, & (x, y) &\mapsto (c^{-1}x, y).\end{aligned}$$

Hence,

- the domain curve is $H: y(1 + ax^2) = cx(1 + dy^2)$
- the codomain curve is $E: ex^2 + y^2 = 1 + fx^2y^2$.

The 2-isogeny map σ' from H to E and its dual $\hat{\sigma}'$ can be derived by

$$\begin{aligned}\sigma' &= (\eta \circ \pi \circ \xi \circ \lambda \circ \epsilon \circ \gamma \circ \phi), \\ \hat{\sigma}' &= (\phi^{-1} \circ \hat{\gamma} \circ \epsilon^{-1} \circ \lambda^{-1} \circ \xi^{-1} \circ \pi^{-1} \circ \eta^{-1})\end{aligned}$$

The following theorem will prove the isogeny map given above.

Theorem 4.3.1. Let $a, c, d, e, f \in \mathbb{K}$, $e = -d$, $f = a/c^2 - d$ and $acd(a - c^2d) \neq 0$

Then, the curve

$$H: y(1 + ax^2) = cx(1 + dy^2)$$

is 2-isogenous over \mathbb{K} to the twisted Edwards curve

$$E_{e,f}: ex^2 + y^2 = 1 + fx^2y^2.$$

The 2-isogeny and its dual are given explicitly as follows:

$$\begin{aligned}\sigma: H &\rightarrow E, & (x, y) &\mapsto \left(\frac{2y}{1 - dy^2}, \frac{1 + ax^2}{1 - ax^2} \right), \\ \hat{\sigma}: E &\rightarrow H, & (x, y) &\mapsto \left(\frac{xy}{c(1 + dx^2)}, \frac{x}{y} \right).\end{aligned}$$

Proof.

- In order E to be an twisted Edwards curve, first thing to show is that the $\Delta_E \neq 0$.

– By the definition of Δ_H ;

$$a, c, d, (a - c^2d) \neq 0 \quad \rightarrow \quad \Delta_E = ef(e - f) = ad(a - c^2d)/c^4 \neq 0.$$

Thus, $E_{e,f}$ defines a twisted Edwards curve

- Secondly, show that σ is an isogeny from H to E .

1. By substituting u with $2y/(1 - dy^2)$ and v with $(1 + ax^2)/(1 - ax^2)$,

it is obtained that

$$eu^2 + v^2 - 1 - fu^2v^2 = \left(y(1 + ax^2) - cx(1 + dy^2) \right) \cdot \frac{4a(y(1 + ax^2) + cx(1 + dy^2))}{-c^2(1 - ax^2)^2(1 - dy^2)^2}.$$

It means that, σ is a rational map from H to E .

2. $\sigma((0, 0)) = (0, 0)$ shows that σ is an isogeny from H to E .

• Next step is to show that $\hat{\sigma}$ is an isogeny from E to H .

1. By substituting u with $xy/(c(1 + dx^2))$ and v with x/y , it is obtained that

$$v(1 + au^2) - cu(1 + dv^2) = \left(ex^2 + y^2 - 1 - fx^2y^2 \right) \cdot \frac{-x}{(1 + dx^2)^2y}.$$

It means that, $\hat{\sigma}$ is a rational map from E to H .

2. $\hat{\sigma}((0, 0)) = (0, 0)$ shows that $\hat{\sigma}$ is an isogeny from E to H .

• By Definition 2.3.2, it must be shown that composition of the isogeny map and its dual gives the multiplication map.

1.

$$\hat{\sigma} \circ \sigma = \left(\frac{2y(1 + ax^2)(1 - dy^2)}{c(1 - ax^2)(1 + dy^2)^2}, \frac{2y(1 - ax^2)}{(1 + ax^2)(1 - dy^2)} \right).$$

By substituting $y(1 + ax^2)$ with $cx(1 + dy^2)$ in the map above, it can be obtained that $\hat{\sigma} \circ \sigma = [2]_H \in \mathbb{K}(H)$.

2. With the same manner above,

- replace $(1 - ex^2)^2 + (f - e)x^2y^2$ with $(1 + dx^2)(y^2 + dx^2)$,
- replace $(1 - ex^2)^2 - (f - e)x^2y^2$ with $(1 + dx^2)(2 - y^2 + dx^2)$

in

$$\sigma \circ \hat{\sigma} = \left(\frac{2xy}{y^2 + ex^2}, \frac{(1 - ex^2)^2 + (f - e)x^2y^2}{(1 - ex^2)^2 - (f - e)x^2y^2} \right).$$

Then, the remaining expression gives $[2]_E \in \mathbb{K}(E)$.

□

4.3.2 2-ISOGENY IN $\mathbb{P}^1 \times \mathbb{P}^1$

The $\mathbb{P}^1 \times \mathbb{P}^1$ embedding of the curve E is

$$\mathcal{E}_{e,f}: \quad eX^2T^2 + Y^2Z^2 = T^2Z^2 + fX^2Y^2,$$

The projective Huff curve \mathcal{H} is 2-isogeneous over \mathbb{K} to \mathcal{E} . The 2-isogeny map and its dual are given as

$$\begin{aligned} ((X: Z), (Y: T)) &\mapsto ((2YT: T^2 - dY^2), (Z^2 + aX^2: Z^2 - aX^2)), \\ ((X: Z), (Y: T)) &\mapsto ((XYZ: cT(Z^2 + dX^2)), (XT: YZ)). \end{aligned}$$

Note that, in order to obtain faster arithmetic, d is set to -1 In Chapter 5.



CHAPTER 5

EFFICIENT IMPLEMENTATION

This chapter includes efficient implementations of point addition and doubling operations on Huff curve. Both operations are shown for various cases such as setting curve constants to special values. Also, 4-way parallel implementation of doubling formula which is obtained by using isogenies and 4-way parallel implementation of mixed addition formula can be found. These implementations are derived for both \mathbb{P}^2 and $\mathbb{P}^1 \times \mathbb{P}^1$ embeddings and the formulas given in 3.3.2 and 3.3.1 are used.

5.1 NOT SO FAST ARITHMETIC ON \mathbb{P}^2 EMBEDDING

In this section, \mathbb{K} is a field with $\text{char}(\mathbb{K}) \neq 2$, $a, d \in \mathbb{K}$, \hat{H} is the homogeneous projective form of extended Huff curve defined over \mathbb{K} and $P_1 = (X_1 : Y_1 : Z_1)$, $P_2 = (X_2 : Y_2 : Z_2)$ are points on curve \hat{H} .

5.1.1 POINT ADDITION ON \mathbb{P}^2 EMBEDDING

Dedicated addition

Recall that the addition of two distinct points P_1 and P_2 equals to $P_3 = (X_3 : Y_3 : Z_3)$ can be calculated as follows:

$$C_0 := X_1 \cdot Z_2, \quad C_1 := Y_1 \cdot Z_2, \quad C_2 := aC_0, \quad C_3 := X_2 \cdot Z_1,$$

$$C_4 := dC_1, \quad C_5 := Y_2 \cdot Z_1,$$

$$R_0 := (Z_1 + a \cdot X_1) \cdot (Z_2 + X_2) - C_2 - C_3, \quad R_1 := (Z_1 + d \cdot Y_1) \cdot (Z_2 + Y_2) - C_4 - C_5,$$

$$R_2 := (C_0 - C_3) \cdot R_1, \quad R_3 := (C_1 - C_5) \cdot R_0,$$

$$X_3 := (C_0 - C_3) \cdot (C_1 + C_5) \cdot R_2, \quad Y_3 := (C_0 + C_3) \cdot (C_1 - C_5) \cdot R_3, \quad Z_3 := R_2 \cdot R_3.$$

- Dedicated addition takes $13\mathbf{M}+4\mathbf{D}+14\mathbf{a}$.
- $a = d = 1$, it can be computed in $13\mathbf{M}+14\mathbf{a}$.
- Dedicated mixed addition costs $11\mathbf{M}+4\mathbf{D}+12\mathbf{a}$.
- Dedicated mixed addition with $a = d = 1$ costs $11\mathbf{M}+12\mathbf{a}$.

Unified addition

The alternative addition formulas of the points P_1 and P_2 , which can be used for the case that $P_1 = P_2$ are represented as

$$\begin{aligned}
C_0 &:= X_1 \cdot X_2, & C_1 &:= Y_1 \cdot Y_2, & C_2 &:= Z_1 \cdot Z_2, & C_3 &:= aC_0, & C_4 &:= dC_1, \\
R_0 &:= (X_1 + Z_1) \cdot (X_2 + Z_2) - C_0 - C_2, & R_1 &:= (Y_1 + Z_1) \cdot (Y_2 + Z_2) - C_1 - C_2, \\
R_2 &:= (C_4 - C_2) \cdot (C_2 + C_3), & R_3 &:= (C_3 - C_2) \cdot (C_2 + C_4), \\
R_4 &:= R_0 \cdot (C_4 - C_2), & R_5 &:= R_1 \cdot (C_3 - C_2), \\
X_3 &:= R_4 \cdot R_2, & Y_3 &:= R_5 \cdot R_3, & Z_3 &:= R_2 \cdot R_3.
\end{aligned}$$

- Unified addition takes $12\mathbf{M}+2\mathbf{D}+14\mathbf{a}$
- $a = d = 1$, it can be computed in $12\mathbf{M}+14\mathbf{a}$.
- Unified mixed addition costs $11\mathbf{M}+2\mathbf{D}+12\mathbf{a}$.
- Unified mixed addition with $a = d = 1$ costs $11\mathbf{M}+12\mathbf{a}$.

5.1.2 POINT DOUBLING ON \mathbb{P}^2 EMBEDDING

The doubling formulas of the point P_1 , $[2]P_1 = (X_3 : Y_3 : Z_3)$ can be calculated as

$$\begin{aligned}
C_0 &:= X_1^2, & C_1 &:= Y_1^2, & C_2 &:= Z_1^2, & C_3 &:= a \cdot C_0, & C_4 &:= d \cdot C_1, \\
R_0 &:= (X_1 + Z_1)^2 - C_0 - C_2, & R_1 &:= (Y_1 + Z_1)^2 - C_1 - C_2, \\
R_2 &:= (C_4 - C_2) \cdot (C_2 + C_3), \\
R_3 &:= (C_3 - C_2) \cdot (C_2 + C_4), & R_4 &:= R_0 \cdot (C_4 - C_2), & R_5 &:= R_1 \cdot (C_3 - C_2), \\
X_3 &:= R_4 \cdot R_2, & Y_3 &:= R_5 \cdot R_3, & Z_3 &:= R_2 \cdot R_3.
\end{aligned}$$

- Doubling takes $7\mathbf{M}+5\mathbf{S}+2\mathbf{D}+11\mathbf{a}$.
- Also, for the case $a = d = 1$, it can be computed in $7\mathbf{M}+5\mathbf{S}+11\mathbf{a}$.

5.2 FASTER ARITHMETIC ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING

In this section, \mathbb{K} is a field with $\text{char}(\mathbb{K}) \neq 2$, $a, c, d \in \mathbb{K}$, \mathcal{H} is the projective closure of extended Huff curve in $\mathbb{P}^1 \times \mathbb{P}^1$ defined over \mathbb{K} and $P_1 = ((X_1 : Z_1), (Y_1 : T_1))$, $P_2 = ((X_2 : Z_2), (Y_2 : T_2))$ are points on curve \mathcal{H} .

5.2.1 POINT ADDITION ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING

Dedicated Addition

- The addition of two distinct points P_1 and P_2 in $\mathbb{P}^1 \times \mathbb{P}^1$ is calculated below:

$$\begin{aligned}
C_0 &:= aX_2, C_0 := Z_2 - C_0, C_1 := dY_2, C_1 := T_2 - C_1, R_0 := Y_1 \cdot T_2, \\
R_1 &:= T_1 \cdot Y_2, R_2 := dR_1, R_2 := R_2 + R_0, T_3 := T_1 - Y_1, T_3 := C_1 \cdot T_3, \\
T_3 &:= T_3 + R_2, Y_3 := R_0 - R_1, R_2 := X_1 \cdot Z_2, R_0 := R_0 + R_1, \\
R_1 &:= Z_1 \cdot X_2, Z_3 := Z_1 - X_1, Z_3 := C_0 \cdot Z_3, Z_3 := Z_3 + R_2, \\
X_3 &:= R_2 - R_1, \mathbf{X}_3 := X_3 \cdot R_0, R_0 := aR_1, R_0 := R_0 + Z_3, \mathbf{Z}_3 := Y_3 \cdot R_0 \\
R_0 &:= R_2 + R_1, \mathbf{Y}_3 := Y_3 \cdot R_0, R_2 := R_2 - R_1, \mathbf{T}_3 := T_3 \cdot R_2.
\end{aligned}$$

– Dedicated addition takes $10\mathbf{M}+4\mathbf{D}+13\mathbf{a}$.

– Dedicated readdition takes $10\mathbf{M}+2\mathbf{D}+11\mathbf{a}$ if $C_0 = Z_2 - aX_2$ and $C_1 = T_2 - dY_2$

- Dedicated mixed addition

$$\begin{aligned}
R_0 &:= X_1 \cdot C_0, R_1 := Z_1 \cdot X_2, R_2 := Y_1 \cdot C_1, R_3 := T_1 \cdot Y_2, R_0 := Z_1 + R_0, \\
R_4 &:= X_1 + R_1, R_2 := T_1 + R_2, R_5 := Y_1 + R_3, R_1 := X_1 - R_1, R_3 := Y_1 - R_3, \\
Z_3 &:= R_0 \cdot R_3, X_3 := R_1 \cdot R_5, T_3 := R_2 \cdot R_1, Y_3 := R_3 \cdot R_4.
\end{aligned}$$

- Dedicated mixed addition takes $8\mathbf{M}+6\mathbf{a}$.
- $C_0 := aX_2$ and $C_1 := dY_2$ is pre-computed and cached. 4-way parallel algorithm when similar operations are grouped, with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$
- Dedicated addition with $a = d = 1$ and $Z_2 = T_2 = 1$ takes $8\mathbf{M}+6\mathbf{a}$. The following is a 4-way parallel algorithm when similar operations are grouped, with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$:

$$\begin{aligned}
R_0 &:= X_1 \cdot X_2, R_1 := Z_1 \cdot X_2, R_2 := Y_1 \cdot Y_2, R_3 := T_1 \cdot Y_2, R_0 := Z_1 + R_0, R_4 := \\
&X_1 + R_1, R_2 := T_1 + R_2, R_5 := Y_1 + R_3, R_1 := X_1 - R_1, R_3 := Y_1 - R_3, \\
&Z_3 := R_0 \cdot R_3, X_3 := R_1 \cdot R_5, T_3 := R_2 \cdot R_1, Y_3 := R_3 \cdot R_4.
\end{aligned}$$

- Dedicated mixed addition takes $8\mathbf{M}+6\mathbf{a}$.
- $C_0 := aX_2$ and $C_1 := dY_2$ is pre-computed and cached. 4-way parallel algorithm when similar operations are grouped, with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$

Unified Addition

- Unified readdition takes $10\mathbf{M}+2\mathbf{D}+10\mathbf{a}$ if C_0 and C_1 are pre-computed and reused in the below algorithm. Unified addition takes $10\mathbf{M}+2\mathbf{D}+12\mathbf{a}$:

$$\begin{aligned}
C_0 &:= X_2 + Z_2, C_1 := Y_2 + T_2, R_0 := X_1 \cdot X_2, R_1 := Y_1 \cdot Y_2, R_2 := \\
&Z_1 \cdot Z_2, R_3 := T_1 \cdot T_2, X_3 := X_1 + Z_1, Y_3 := Y_1 + T_1, \\
&R_4 := R_0 + R_2, R_5 := R_1 + R_3, \\
X_3 &:= X_3 \cdot C_0, Y_3 := Y_3 \cdot C_1, Z_3 := aR_0, T_3 := dR_1, R_0 := R_3 + T_3, T_3 := \\
&R_3 - T_3, R_1 := R_2 + Z_3, Z_3 := R_2 - Z_3, X_3 := X_3 - R_4, Y_3 := Y_3 - R_5, \\
X_3 &:= X_3 \cdot T_3, Y_3 := Y_3 \cdot Z_3, Z_3 := Z_3 \cdot R_0, T_3 := T_3 \cdot R_1.
\end{aligned}$$

- Unified readdition with $Z_2 = T_2 = 1$, takes $8\mathbf{M}+6\mathbf{a}$ if $C_0 := aX_2$ and $C_1 := dY_2$ is pre-computed and cached. The following is a 4-way parallel algorithm when similar operations are grouped, with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$:

$$\begin{aligned}
R_0 &:= X_1 \cdot C_0, & R_1 &:= Z_1 \cdot X_2, & R_2 &:= Y_1 \cdot C_1, & R_3 &:= T_1 \cdot Y_2, & R_4 &:= Z_1 + R_0, \\
R_1 &:= X_1 + R_1, & R_5 &:= T_1 + R_2, & R_3 &:= Y_1 + R_3, & R_0 &:= Z_1 - R_0, & R_2 &:= T_1 - R_2, \\
Z_3 &:= R_0 \cdot R_5, & X_3 &:= R_1 \cdot R_2, & T_3 &:= R_2 \cdot R_4, & Y_3 &:= R_3 \cdot R_0.
\end{aligned}$$

- Unified addition with $a = d = 1$ and $Z_2 = T_2 = 1$ takes $8\mathbf{M}+6\mathbf{a}$. The following is a 4-way parallel algorithm with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$:

$$\begin{aligned}
R_0 &:= X_1 \cdot X_2, & R_1 &:= Z_1 \cdot X_2, & R_2 &:= Y_1 \cdot Y_2, & R_3 &:= T_1 \cdot Y_2, \\
R_4 &:= Z_1 + R_0, & R_1 &:= X_1 + R_1, & R_5 &:= T_1 + R_2, & R_3 &:= Y_1 + R_3, \\
R_0 &:= Z_1 - R_0, & \textit{idle} & & R_2 &:= T_1 - R_2, & \textit{idle} & \\
Z_3 &:= R_0 \cdot R_5, & X_3 &:= R_1 \cdot R_2, & T_3 &:= R_2 \cdot R_4, & Y_3 &:= R_3 \cdot R_0.
\end{aligned}$$

5.2.2 POINT DOUBLING ON $\mathbb{P}^1 \times \mathbb{P}^1$ EMBEDDING

- Doubling takes $4\mathbf{M}+6\mathbf{S}+2\mathbf{D}+10\mathbf{a}$:

$$\begin{aligned}
R_0 &:= X_1^2, & X_3 &:= X_1 + Z_1, & X_3 &:= X_3^2, & X_3 &:= X_3 - R_0, & Z_3 &:= Z_1^2, & X_3 &:= \\
& & X_3 - Z_3, & R_0 &:= aR_0, & R_1 &:= Z_3 + R_0, & Z_3 &:= Z_3 - R_0, & R_0 &:= Y_1^2, \\
& & Y_3 &:= Y_1 + T_1, & Y_3 &:= Y_3^2, & Y_3 &:= Y_3 - R_0, & T_3 &:= T_1^2, \\
& & Y_3 &:= Y_3 - T_3, & R_0 &:= dR_0, & R_2 &:= T_3 + R_0, & T_3 &:= T_3 - R_0, \\
& & X_3 &:= X_3 \cdot T_3, & Y_3 &:= Y_3 \cdot Z_3, & Z_3 &:= Z_3 \cdot R_2, & T_3 &:= T_3 \cdot R_1.
\end{aligned}$$

- Doubling exploiting the 2-isogeny decomposition, shown in (4.1) with $a = d$, takes $8\mathbf{M}+2\mathbf{D}+8\mathbf{a}$:

$$\begin{aligned}
R_0 &:= aX_1, & X_3 &:= X_1 \cdot T_1, & R_0 &:= R_0 \cdot Y_1, & T_3 &:= T_1 \cdot Z_1, \\
Y_3 &:= Y_1 \cdot Z_1, & Z_3 &:= T_3 + R_0, & T_3 &:= T_3 - R_0, & R_0 &:= X_3 + Y_3, & Y_3 &:= X_3 - Y_3, \\
X_3 &:= aR_0, & X_3 &:= X_3 \cdot Y_3, & R_0 &:= R_0 \cdot T_3, & T_3 &:= T_3 \cdot Z_3, & Y_3 &:= Y_3 \cdot Z_3, \\
Z_3 &:= T_3 - X_3, & T_3 &:= T_3 + X_3, & X_3 &:= R_0 + Y_3, & Y_3 &:= R_0 - Y_3.
\end{aligned}$$

- Doubling with $a = d = 1$ via 2-isogeny decomposition shown in (4.1) takes $8\mathbf{M}+8\mathbf{a}$. The following is a 4-way parallel algorithm with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$:

$$\begin{aligned}
R_0 &:= X_1 \cdot T_1, & R_1 &:= Y_1 \cdot Z_1, & R_2 &:= T_1 \cdot Z_1, & R_3 &:= X_1 \cdot Y_1, \\
X_3 &:= R_0 + R_1, & Y_3 &:= R_0 - R_1, & T_3 &:= R_2 - R_3, & Z_3 &:= R_2 + R_3, \\
R_0 &:= X_3 \cdot T_3, & R_1 &:= Y_3 \cdot Z_3, & R_2 &:= T_3 \cdot Z_3, & R_3 &:= X_3 \cdot Y_3, \\
X_3 &:= R_0 + R_1, & Y_3 &:= R_0 - R_1, & Z_3 &:= R_2 - R_3, & T_3 &:= R_2 + R_3.
\end{aligned}$$

5.2.3 POINT ADDITION AND DOUBLING FOR CO-FACTOR 4

Unified addition

Unified addition with $a = d = 2$ and $Z_2 = T_2 = 1$ takes $8\mathbf{M}+8\mathbf{a}$. The following is a 4-way parallel algorithm with cost $4 \times (2\mathbf{M} + 2\mathbf{a})$:

$$\begin{aligned}
R_0 &:= X_1 \cdot X_2, & R_1 &:= Z_1 \cdot X_2, & R_2 &:= Y_1 \cdot Y_2, & R_3 &:= T_1 \cdot Y_2, \\
R_0 &:= R_0 + R_0, & R_1 &:= R_1 + X_1, & R_2 &:= R_2 + R_2, & R_3 &:= R_3 + Y_1, \\
R_4 &:= R_0 + Z_1, & R_7 &:= R_0 - Z_1, & R_6 &:= R_2 - T_1, & R_5 &:= R_2 + T_1, \\
T_3 &:= R_4 \cdot R_6, & Y_3 &:= R_7 \cdot R_3, & X_3 &:= R_6 \cdot R_1, & Z_3 &:= R_5 \cdot R_7.
\end{aligned}$$

Doubling

Doubling with $a = d = 2$ via 2-isogeny decomposition shown in (4.1) takes $8\mathbf{M}+10\mathbf{a}$. The following is a 4-way parallel algorithm with cost $4 \times (2\mathbf{M} + 4\mathbf{a})$:

<i>idle</i>	<i>idle</i>	<i>idle</i>	$R_4 := X_1 + X_1$
$R_0 := X_1 \cdot T_1,$	$R_1 := Y_1 \cdot Z_1,$	$R_2 := T_1 \cdot Z_1,$	$R_3 := R_4 \cdot Y_1,$
$X_3 := R_0 + R_1,$	$Y_3 := R_0 - R_1,$	$T_3 := R_2 - R_3,$	$Z_3 := R_2 + R_3,$
<i>idle</i>	<i>idle</i>	<i>idle</i>	$R_4 := X_3 + X_3$
$R_0 := X_3 \cdot T_3,$	$R_1 := Y_3 \cdot Z_3,$	$R_2 := T_3 \cdot Z_3,$	$R_3 := R_4 \cdot Y_3,$
$X_3 := R_0 + R_1,$	$Y_3 := R_0 - R_1,$	$Z_3 := R_2 - R_3,$	$T_3 := R_2 + R_3.$





CHAPTER 6

COMPARISON AND CONCLUSION

Huff curve form was the slowest curve model in terms of group operations. So, it is more sensible to compare efficiency of extended Huff curve with older Huff forms, and also with the other curve forms. Because, eventhough the Huff form does not become the fastest curve model, the efficiency improvement of Huff curve itself is remarkable. Furthermore, it can get desirable for parallel implementations, so that Huff curve is almost as fast as twisted Edwards curve in 4-way parallel environments. The presented formulas are expected to be attractive for parallel processing (e.g. special hardware, SIMD, video card settings). There is a need for further investigation in this direction.

The first section makes comparison between Huff forms of elliptic curve. The second section provides the detailed comparison between different curve models. Two look up tables for operation counts of different curve models are given. And, there is also two tables which carry the cost estimates for 1-NAF and 4-NAF variable point scalar multiplication for each model. Finally, parallel implementation assessment is made.

6.1 COMPARISON BETWEEN HUFF FORMS

As stated before, the speed oriented group operation algorithms of Huff curve were embedded into \mathbb{P}^2 in the previous works. In this work, extended Huff curve is embedded into $\mathbb{P}^1 \times \mathbb{P}^1$, and it solidly provides lower total degree formulas for each coordinate. These formulas set the new operation count record for each group operation in Huff form. Table 6.1 compares the results of Chapter 5 for

DBL, **muADD**, and **uADD**¹ with the literature.

Table 6.1: Speed oriented operation counts for Huff form

Source & the curve equation	h	DBL	muADD	uADD
(Wu & Feng, 2012) $b = 1$, $X(aY^2 - Z^2) = Y(X^2 - Z^2)$	4	$6\mathbf{M}+5\mathbf{S}+1\mathbf{D}+12\mathbf{a}$	$10\mathbf{M}+1\mathbf{D}+14\mathbf{a}$	$11\mathbf{M}+1\mathbf{D}+14\mathbf{a}$
(Joye et al., 2010), $aX(Y^2 - Z^2) = bY(X^2 - Z^2)$	8	$6\mathbf{M}+5\mathbf{S}+13\mathbf{a}$	$10\mathbf{M}+14\mathbf{a}$	$11\mathbf{M}+14\mathbf{a}$
This work , $YT(Z^2 + 2X^2) = cXZ(T^2 + 2Y^2)$	4	$8\mathbf{M}+10\mathbf{a}$ $4 \times (2\mathbf{M}+4\mathbf{a})$	$8\mathbf{M}+8\mathbf{a}$ $4 \times (2\mathbf{M}+2\mathbf{a})$	$10\mathbf{M}+14\mathbf{a}$
This work , $YT(Z^2 + X^2) = cXZ(T^2 + Y^2)$	8	$8\mathbf{M}+8\mathbf{a}$ $4 \times (2\mathbf{M}+2\mathbf{a})$	$8\mathbf{M}+6\mathbf{a}$ $4 \times (2\mathbf{M}+2\mathbf{a})$	$10\mathbf{M}+12\mathbf{a}$

- The column h represents the least possible cofactor in the given curve model.

Note that, in the last two entries of Table 6.1, an additional condition $T_2 = 1$ for **muADD** is underlined, since $\mathbb{P}^1 \times \mathbb{P}^1$ embedding is used. Also, it is assumed that, $2x^2 + 1 \in \mathbb{K}[x]$ is irreducible in the case of $YT(Z^2 + 2X^2) = cXZ(T^2 + 2Y^2)$. This condition can be obtained from (Joye et al., 2010, Section 3.3). Since the counts of addition/subtraction (**a**) were not specified in the reference works, they are stated in the Table 6.1 without eliminating common subexpressions. Eventhough, the formulas for the extended Huff curve do not require moving the identity to a point at infinity, selecting the identity element as $(0: 1: 0)$ gives the best operation counts for the previous forms, which are reported in reference

¹**DBL**, **mADD**, **muADD**, **ADD**, **uADD** represent doubling, mixed addition, mixed unified addition where $Z_2 = 1$, addition and unified addition respectively.

works. In (Joye et al., 2010), operation counts for twisted Huff curve which is written in the form $aX(Y^2 - dZ^2) = bY(X^2 - dZ^2)$ are not provided, thus Table 6.1 does not contain this case.

As it is seen in the table, this work provides 2M faster formulas for both **DBL**, **muADD** and 1M faster formulas for **uADD** than the fastest so far formulas given in (Joye et al., 2010).

6.2 COMPARISON BETWEEN OTHER FORMS OF ELLIPTIC CURVES

It is shown that extended Huff curve is much more efficient than the previous Huff forms. Nevertheless, Huff form is not fast enough in sequential implementations compared to others. It can get competitive with Short Weierstrass and twisted Hessian forms under some conditions. Yet, extended Huff curve can be preferable for parallel applications, thanks to the 4-way parallel formulas for doubling and mixed-addition given in Chapter 5.

6.2.1 GROUP OPERATION COSTS FOR DIFFERENT CURVE FORMS

The following two tables indicates operation counts of **DBL**, **mADD**, **muADD**, **ADD**, **uADD** for the different forms of elliptic curves. The operation counts of Jacobi quartic, Jacobi intersection, twisted Hessian, twisted Edwards are obtained from (Hisil, Wong, Carter, & Dawson, 2009), (Hisil, 2010b), (Bernstein et al., 2015)-(Hisil, 2010b), (Hisil et al., 2008) respectively. While Table 6.2 gives operation counts for normal case, Table 6.3 gives operation counts of each curve form with the case where the curve constants are set to small elements such as $-1, 1 \dots$ etc.

Table 6.2: Operation Counts for Different Curve Forms

Form	DBL	mADD/muADD	ADD/uADD
Huff	4M+6S+2D+10a	8M+ 2D+ 6a	10M+4D+12a
		8M+ 2D+10a	10M+2D+12a
Jacobi Q.	2M+5S+1D+ 8a	6M+3S+ 3D+21a	6M+4S+3D+21a
		6M+3S+ 5D+23a	7M+4S+4D+19a
Jacobi I.	2M+5S+2D+ 8a	10M+1D+ 9a	11M+1D+ 9a
		10M+ 2S+5D+13a	11M+2S+5D+13a
Hessian	6M+3S+1D+ 3a	10M+1D+13a	12M+1D+ 3a
		10M+1D+13a	12M+1D+ 3a
Edwards	3M+4S+1D+ 7a	8M+1D+ 6a	9M+1D+ 6a
		8M+2D+ 6a	9M+2D+ 6a

Table 6.3: Operation Counts with Special Conditions for Different Curve Forms

Form & Condition	DBL	mADD/muADD	ADD/uADD
Huff $a = d = 1$	8M+ 8a	8M+ 2D+ 6a	10M+4D+12a
		8M+ 6a	10M+2D+12a
Jacobi Q. $a = -1/2$	2M+5S+ 7a	6M+ 3S+1D+19a	6M+4S+2D+21a
		6M+ 3S+2D+21a	7M+4S+3D+19a
Jacobi I. $b = 1$	2M+5S+1D+ 7a	10M+ 9a	11M+ 9a
		10M+ 1S+2D+15a	11M+1S+2D+15a
Hessian $a = 1$	6M+2S+1D+ 3a	9M+17a	11M+17a
		9M+17a	11M+17a
Edwards $a = -1$	3M+4S+1D	7M+ 8a	8M+ 8a
		7M+1D+ 8a	8M+1D+ 8a

- The curve equations are taken as follows in the above tables.
 - **Extended Huff** : $y(1 + ax^2) = cx(1 + dy^2)$,
 - **Jacobi Quartic** : $y^2 = dx^4 + 2ax^2 + 1$,
 - **Jacobi Intersection** : $as^2 + c^2 = 1, ds^2 + d^2 = 1$,
 - **Twisted Hessian** : $ax^3 + y^3 + 1 = dxy$,
 - **Twisted Edwards** : $ax^2 + y^2 = 1 + dx^2y^2$.

6.2.2 SCALAR MULTIPLICATION COSTS

Single-scalar variable-point multiplication is the operation that this work focuses on. Therefore, this subsection contains the comparison of scalar multiplication costs between different curve models. Signed sliding window algorithm is the best method to make this comparison. Table 6.4 provides 1-NAF (non-adjacent form) and Table 6.5 provides 4-NAF costs of each curve form. Note that numbers 1 and 4 stands for the window sizes in the mentioned algorithm.

The cost estimation method, given in (Bernstein & Lange, 2007, Section 6) is used. The costs are deduced for each curve model by using records from Table 6.3 which includes operation counts for the best case of each curve model. Thus, the fastest values are chosen. The columns (1,1), (.8,.5), and (.8,0) in the Table 6.4 and Table 6.5 shows different **S/M** and **D/M** values, respectively.

Table 6.4: Cost estimates for 1-NAF variable point scalar multiplication

Curve model	h	cost per scalar bit			cost for 256 bit scalar		
		(1,1)	(.8,.5)	(.8,0)	(1, 1)	(.8,.5)	(.8,0)
Huff, (Joye et al., 2010)	4	15.67	14.00	13.33	4011	3584	3413
Huff $a = d = 2$ <i>this work</i>	4	10.67	10.67	10.67	2731	2731	2731
Hessian , $a = \pm 1$ (Hisil, 2010b), (Bernstein et al., 2015)	3	11.00	10.80	10.60	2816	2765	2714
Weierstrass $a = -3$	1	11.67	10.40	10.40	2987	2662	2662
Jacobi Intersection , $b = 1$ (Hisil, 2010b)	4	10.33	9.53	9.33	2645	2441	2389
Jacobi Quartic , $a = -1/2$ (Hisil et al., 2009)	2	10.33	8.97	8.80	2645	2295	2253
Twisted Edwards , $a = -1$ (Hisil et al., 2008)	4	9.33	8.53	8.53	2389	2185	2185

Table 6.5: Cost estimates for sliding window 4-NAF variable point scalar multiplication

Curve model	h	cost per scalar bit			cost for 256 bit scalar		
		(1,1)	(.8,.5)	(.8,0)	(1, 1)	(.8,.5)	(.8,0)
Huff, (Joye et al., 2010)	4	14.09	12.52	11.93	3608	3206	3055
Huff $a = d = 2$ <i>this work</i>	4	9.75	9.75	9.75	2496	2496	2496
Hessian , $a = \pm 1$ (Hisil, 2010b), (Bernstein et al., 2015)	3	9.94	9.75	9.55	2546	2496	2445
Weierstrass $a = -3$	1	10.51	9.37	9.37	2690	2399	2399
Jacobi Intersection , $b = 1$ (Hisil, 2010b)	4	9.16	8.29	8.00	2344	2121	2049
Jacobi Quartic , $a = -1/2$ (Hisil et al., 2009)	2	8.99	7.79	7.69	2301	1994	1970
Twisted Edwards , $a = -1$ (Hisil et al., 2008)	4	8.40	7.62	7.62	2152	1950	1950

The calculation of n -bit scalar multiplication costs n doublings and $n/3$ mixed additions when 1-NAF algorithm is used.

When 4-NAF is used, it costs $n - 4.5$ doublings, $7n/48 + 5.2$ readditions, $b/48 + 0.9$ mixed additions, and 0.9 non-mixed additions.

Tables are ordered by the values of the column (.8,0). For this case Huff form saves its place in the order. But, efficiency increase of Huff curve itself is obvious, it is in a range from 1.22x to 1.47x.

Huff curve can also get competitive with twisted Hessian and short Weierstrass curves in single-base scalar multiplications depending on the **S/M** and **D/M** values. Also, double-base scalar multiplication cost estimates for twisted Hessian curves in (Bernstein et al., 2015) must be considered.

In the sequential implementations, especially in windowed scalar multiplica-

tions, Huff curve cannot be faster than the \mathbb{P}^3 embeddings of twisted Edwards, Jacobi quartic, and Jacobi intersection curves. However, Huff form is very competitive in 4-way parallel environments.

6.2.3 COMPARISON FOR 4-WAY PARALLEL SETTING

The most common group operations, doubling and mixed addition, are shown in Chapter 5, are to be 4-way parallelizable for extended Huff curve. Thus, it is tempting to compare the performance of the new formulas with the fastest formulas developed for twisted Edwards curves for which efficient 4-way parallel algorithms are given in (Hisil et al., 2008). The following table states the operation counts of both twisted Edwards and extended Huff curves for 4-way parallel setting.

Table 6.6: Comparison between twisted Edwards and extended Huff curves for 4-way parallel setting

Curve model	h	DBL	muADD
Extended Huff, $a = d = 2$	4	$4 \times (2\mathbf{M})$	$4 \times (2\mathbf{M})$
Twisted Edwards, $a = -1$	4	$4 \times (1\mathbf{M} + 1\mathbf{S})$	$4 \times (2\mathbf{M})$

Both forms should give similar performance if $\mathbf{M} = \mathbf{S}$. The fastest 4-way parallel mixed addition takes $4 \times 2\mathbf{M}$ in both forms. Both curve models are expected to give similar performances when double-and-add or 1-NAF scalar multiplication algorithm is used where $\mathbf{M} = \mathbf{S}$. Huff form is slower yet close in performance when the window size is chosen to be greater than 1. Because, 4-way parallel full addition on twisted Edwards curve takes $4 \times (2\mathbf{M})$, but Huff slows down to $4 \times (3\mathbf{M})$.

There is another advantage of twisted Edwards curves. Conversion of a projective point is slightly faster for the embedding used by twisted Edwards

curve. The conversion of a projective point $(X : Y : T : Z)$ to the affine point $(X/Z, Y/Z)$ takes $\mathbf{I} + 2\mathbf{M}$. The conversion of a projective point $((X : Z), (Y : T))$ on an extended Huff curve to the affine point $(X/Z, Y/T)$ takes $\mathbf{I} + 5\mathbf{M}$ using Montgomery's simultaneous inversion technique (Hankerson et al., 2003). Since \mathbf{I} is many times more costly than \mathbf{M} , the performance difference is not remarkable. $\mathbb{P}^1 \times \mathbb{P}^1$ embedding of twisted Edwards curves and explicit group law formulas are given in (Bernstein & Lange, 2011) in rather a different context, without operation counts. Therefore, it is not clear how the two compare with each other.



References

- Bernstein, D. J., Birkner, P., Joye, M., Lange, T., & Peters, C. (2008). Twisted Edwards curves. In *AFRICACRYPT 2008 proceedings* (Vol. 5023, pp. 389–405). Springer.
- Bernstein, D. J., Chuengsatiansup, C., Kohel, D., & Lange, T. (2015). Twisted Hessian curves. In *Progress in cryptology LATINCRYPT 2015 proceedings* (Vol. 9230, pp. 269–294). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-22174-8_15 doi: 10.1007/978-3-319-22174-8_15
- Bernstein, D. J., & Lange, T. (2007). Faster addition and doubling on elliptic curves. In *ASIACRYPT 2007* (Vol. 4833, pp. 29–50). Springer.
- Bernstein, D. J., & Lange, T. (2011). A complete set of addition laws for incomplete Edwards curves. *Journal of Number Theory*, 131(5), 858–872. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0022314X10002155> doi: <http://dx.doi.org/10.1016/j.jnt.2010.06.015>
- Bernstein, D. J., & Lange, T. (2017). *Explicit-formulas database*. <http://www.hyperelliptic.org/EFD>.
- Bézout, E. (1779). *Théorie générale des équations algébriques; par m. bézout...* de l'imprimerie de Ph.-D. Pierres, rue S. Jacques.
- Billet, O., & Joye, M. (2003). The Jacobi model of an elliptic curve and side-channel analysis. In *Applied algebra, algebraic algorithms and error-correcting codes: 15th international symposium, aecc-15, toulouse, france, may 12-16, 2003 proceedings* (Vol. 2643, pp. 34–42). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-44828-4_5 doi: 10.1007/3-540-44828-4_5
- Bosma, W., Cannon, J., & Playoust, C. (1997). The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4), 235–265. Retrieved from <http://dx.doi.org/10.1006/jsco.1996.0125> (Computational algebra and number theory (London, 1993)) doi: 10.1006/jsco.1996.0125

- Brier, E., & Joye, M. (2002). Weierstraß elliptic curves and side-channel attacks. In *International workshop on public key cryptography* (pp. 335–345).
- Ciss, A. A., & Sow, D. (2011). *On a new generalization of Huff curves*. Cryptology ePrint Archive, Report 2011/580. (<http://eprint.iacr.org/2011/580>)
- Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., & Vercauteren, F. (2005). *Handbook of elliptic and hyperelliptic curve cryptography*. United Kingdom: Chapman and Hall/CRC.
- Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, 22(6), 644–654.
- Digital signature standard (DSS)* (Tech. Rep.). (2000). National Institute of Standards and Technology (NIST).
- Doche, C., Icart, T., & Kohel, D. R. (2006). Efficient scalar multiplication by isogeny decompositions. In *Public key cryptography* (Vol. 3958, pp. 191–206).
- Feng, R., Nie, M., & Wu, H. (2010). Twisted jacobi intersections curves. In *TAMC* (pp. 199–210).
- Hankerson, D., Menezes, A. J., & Vanstone, S. A. (2003). *Guide to elliptic curve cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
- Hisil, H. (2010a). *ecfp256: A crypto_dh application for SUPERCOP*. Retrieved 2017-07-20, from <http://hhisil.yasar.edu.tr/>
- Hisil, H. (2010b). *Elliptic curves, group law, and efficient computation* (Unpublished doctoral dissertation). Queensland University of Technology.
- Hisil, H., Wong, K. K.-H., Carter, G., & Dawson, E. (2008). Twisted Edwards curves revisited. In *ASIACRYPT 2008 proceedings* (Vol. 5350, pp. 326–343). Springer.
- Hisil, H., Wong, K. K.-H., Carter, G., & Dawson, E. (2009). Jacobi quartic curves revisited. In *ACISP 2009 proceedings* (Vol. 5594, pp. 452–468). Springer.
- Huff, G. B. (1948, 06). Diophantine problems in geometry and elliptic ternary forms. *Duke Mathematical Journal*, 15(2), 443–453. Retrieved from

<http://dx.doi.org/10.1215/S0012-7094-48-01543-9> doi: 10.1215/
S0012-7094-48-01543-9

Inc., W. M. (2008). *Maple 12*. <http://www.maplesoft.com/>.

Information technology - security techniques - digital signatures with appendix - part 3: logarithm based mechanisms (Tech. Rep.). (2006). International Organization for Standards (ISO).

Jacobi, C. G. J. (1829). *Fundamenta nova theoriae functionum ellipticarum. auctore d. carolo gustavo iacobo iacobi..* sumtibus fratrum Borntæger.

Joye, M., & Quisquater, J. J. (2001). Hessian elliptic curves and side-channel attacks. In *CHES* (Vol. 2001, pp. 402–410).

Joye, M., Tibouchi, M., & Vergnaud, D. (2010). Huff's model for elliptic curves. In *Algorithmic number theory: 9th international symposium, antis-ix, nancy, france, july 19-23, 2010 proceedings* (Vol. 6197, pp. 234–250). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-14518-6_20 doi: 10.1007/978-3-642-14518-6_20

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177), 203–209.

Kohel, D. (1996). *Endomorphism rings of elliptic curves over finite fields* (Unpublished doctoral dissertation). University of California, Berkley.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Miller, V. S. (1985). Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques* (pp. 417–426). Springer, Berlin, Heidelberg.

Moody, D., & Shumow, D. (2001). *Analogues of Velu's formulas for isogenies on alternate models of elliptic curves*. Cryptology ePrint Archive, Report 2011/430. (<http://eprint.iacr.org/2011/430>)

Odlyzko, A. M. (1984). Discrete logarithms in finite fields and their cryptographic significance. In *Workshop on the theory and application of of cryptographic*

- techniques* (pp. 224–314). Springer, Berlin, Heidelberg.
- Pohlig, S., & Hellman, M. (1978). An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (corresp.). *IEEE Transactions on information Theory*, *24*(1), 106–110.
- Pollard, J. M. (1978). Monte carlo methods for index computation (mod p). *Mathematics of computation*, *32*(143), 918–924.
- Public key cryptography for the financial services industry, key agreement and key transport using elliptic curve cryptography* (Tech. Rep.). (2001). American National Standards Institute (ANSI).
- Public key cryptography for the financial services industry - the elliptic curve digital signature algorithm (ECDSA)* (Tech. Rep.). (2005). Accredited Standards Committee X9, Incorporated.
- Silverman, J. H. (2009). *The arithmetic of elliptic curves* (Vol. 106). Springer Science & Business Media.
- Smart, N. P. (1999). The discrete logarithm problem on elliptic curves of trace one. *Journal of cryptology*, *12*(3), 193–196.
- Smart, N. P. (2001). The hessian form of an elliptic curve. In *CHES* (Vol. 2162, pp. 118–125).
- Standard Specifications For Public-Key Cryptography* (Tech. Rep.). (2000). Institute of Electrical and Electronics Engineers (IEEE).
- Vélu, J. (1971). Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris Sér. AB*, *273*(A238-A241), 238–241.
- Washington, L. C. (2008). *Elliptic curves: number theory and cryptography*. CRC press.
- Wu, H., & Feng, R. (2012). Elliptic curves in Huff’s model. *Wuhan University Journal of Natural Sciences*, *17*(6), 473–480. Retrieved from <http://dx.doi.org/10.1007/s11859-012-0873-9> doi: 10.1007/s11859-012-0873-9

APPENDIX 1- C CODE OF SCALAR MULTIPLICATION

This appendix provides the C code of Sliding window scalar multiplication on extended Huff curve which is adapted from (Hisil, 2010a).

Main Loop includes the code of the main operation which calls the scalar multiplication function and calculates the operation counts for 256-bit base points. In *Group Operations* part, the C codes of sliding window scalar multiplication, each group operation that is used in scalar multiplication on extended Huff curve and next window functions are provided. The window size is set to 5, because it is the optimal window size for extended Huff curve.

MAIN LOOP

```
1 #include <stdio.h>
2 #include "crympix.h"
3 #include "finite256.h"
4 #include "ec_fp_256h.h"
5
6 #ifdef TEST_BOX
7 #include <stdlib.h>
8 #include <math.h>
9 #include <sys/time.h>
10 #include "test.h"
11 #include "timer.h"
12 #else
13 #include <string.h>
14 #include <stdarg.h>
15 #include <math.h>
16 #endif
17
18 #define SECRETKEY_BYTES 32
19 #define PUBLICKEY_BYTES 64
20 #define SHAREDSECRET_BYTES 32
21
22 #ifdef TEST_BOX
23 static
24 #endif
25 int crypto_dh_keypair(unsigned char* pk, unsigned char *sk)
26 {
27     unsigned char zzn[32];
28     unsigned char ttn[32];
29     EC_FP_SMULBASE_256H_CACHE;
30     unsigned int i;
31
32     /*Secret key (To be replaced with a secure pseudorandom number generator).*/
33     for (i = 0; i < SECRETKEY_BYTES; i++) sk[i] = rand();
34
35     #if EC_FP_SMULBASE_SLICE == 0
36     ec_fp_smul_256h_u((uni)pk, (uni)zzn, (uni)(pk + (PUBLICKEY_BYTES/2)), (uni)ttn, (uni)sk, (uni)xn0, (uni)
        yn0);
37     #else
38     #endif
39 #endif
40
41     /*Normalization.*/
42     fp_inv_256((uni)zzn, (uni)zzn);
43     fp_inv_256((uni)ttn, (uni)ttn);
44     fp_mul_256((uni)pk, (uni)pk, (uni)zzn);
45     fp_mul_256((uni)(pk + (PUBLICKEY_BYTES/2)), (uni)(pk + (PUBLICKEY_BYTES/2)), (uni)ttn);
46
47     return 0;
48 }
49
50 #ifdef TEST_BOX
51 static
52 #endif
53 int crypto_dh(unsigned char *s, const unsigned char* pk, const unsigned char *sk){
54     uni_t zzn[FP_LEN], yyn[FP_LEN], ttn[FP_LEN];
55     unsigned char result[32];
56
57     /*scalar multiplication*/
58     ec_fp_smul_256h_u((uni)result, zzn, yyn, ttn, (uni)sk, (uni)pk, (uni)(pk + (PUBLICKEY_BYTES/2)));
```

```

59
60 /*normalization*/
61 fp_inv_256((uni)zsn, (uni)zsn);
62 fp_mul_256((uni)result, (uni)result, (uni)zsn);
63
64 memcpy(s,result,32);
65
66 return 0;
67 }
68
69 #ifdef TEST_BOX
70 static
71 #endif
72 int copyrightclaims(){
73     return 0;
74 }
75
76 #ifdef TEST_BOX
77 static
78 #endif
79 int timingattacks(){
80     return 100;
81 }
82
83 #ifdef TEST_BOX
84 static
85 #endif
86 int patentclaims(){
87     return 0;
88 }
89 void main()
90 {
91     unsigned char pk1[PUBLICKEY_BYTES], sk1[SECRETKEY_BYTES];
92     unsigned char pk2[PUBLICKEY_BYTES], sk2[SECRETKEY_BYTES];
93     unsigned char ss1[PUBLICKEY_BYTES], ss2[SECRETKEY_BYTES];
94     long count, i;
95
96     COUNT_MLD=0;
97     COUNT_ADD=0;
98     COUNT_MUL=0;
99     COUNT_SQR=0;
100
101     printf("\nExtenden Huff (a=d=1) H (fixedbase)\n");
102
103     for(count = 0; count < 10000; count++){
104         crypto_dh_keypair(pk1, sk1);
105         crypto_dh_keypair(pk2, sk2);
106         crypto_dh(ss1, pk1, sk2);
107         crypto_dh(ss2, pk2, sk1);
108
109         for(i = 0; i < 32; i++){
110             if(ss1[i] != ss2[i]){
111                 printf("Error! Secret does not match. (@ %lu)\n", count);
112                 exit(1);
113             }
114         }
115         if((count%10000) == 0){
116             printf("%lu\n", count);
117         }
118     }
119
120     printf("count_COUNT_MLD %lf\n", (double)COUNT_MLD/40000);
121     printf("count_COUNT_ADD %lf\n", (double)COUNT_ADD/40000);
122     printf("count_COUNT_MUL %lf\n", (double)COUNT_MUL/40000);
123     printf("count_COUNT_SQR %lf\n", (double)COUNT_SQR/40000);
124
125
126     copyrightclaims();
127     timingattacks();
128     patentclaims();
129 }

```

algorithm 6.1: Codes/p1Huff/try_ec_fp_256h.c

GROUP OPERATIONS

```

1  /**
2  * Scalar multiplication on extended Huff curves with a = d = 1.
3  *
4  *    $d*Y*T*(Z^2+a*X^2)=c*X*Z*(T^2+b*Y^2)$ .
5  *
6  */
7  #include <stdio.h>
8  #include "crympix.h"
9  #include "finite256.h"
10 #include "ec_fp_256h.h"
11
12 #define WINDOW_SIZE_LTR 5
13
14 #define TABLE_SIZE_LTR (1 << (WINDOW_SIZE_LTR - 2))
15
16 static void ec_fp_cpy_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1, const
    uni T1){
17     fp_cpy_256(X3, X1);
18     fp_cpy_256(Y3, Y1);
19     if(Z1 == NULL && T1 == NULL){
20         fp_set_1_256(Z3, 1);
21         fp_set_1_256(T3, 1);
22     }
23     else{
24         fp_cpy_256(Z3, Z1);
25         fp_cpy_256(T3, T1);
26     }
27 }
28
29 static void ec_fp_neg_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1, const
    uni T1){
30     fp_sub_2_256(X3, 0, X1);
31     fp_sub_2_256(Y3, 0, Y1);
32     if(Z1 == NULL && T1 == NULL){
33         fp_set_1_256(Z3, 1);
34         fp_set_1_256(T3, 1);
35     }
36     else{
37         fp_cpy_256(Z3, Z1);
38         fp_cpy_256(T3, T1);
39     }
40 }
41
42 /*Cost: 5M + 8a.*/
43 static void ec_fp_mdbl_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Y1){
44     uni_t r0[FP_LEN], r1[FP_LEN], r2[FP_LEN], r3[FP_LEN], r4[FP_LEN];
45
46     fp_mul_256(r4, X1, Y1);
47     fp_add_256(X3, X1, Y1);
48     fp_sub_256(Y3, X1, Y1);
49     fp_sub_2_256(T3, 1, r4);
50     fp_add_1_256(Z3, r4, 1);
51
52     fp_mul_256(r0, X3, T3);
53     fp_mul_256(r1, Y3, Z3);
54     fp_mul_256(r2, T3, Z3);
55     fp_mul_256(r3, X3, Y3);
56
57     fp_add_256(X3, r0, r1);
58     fp_sub_256(Y3, r0, r1);
59     fp_sub_256(Z3, r2, r3);
60     fp_add_256(T3, r2, r3);
61
62 }
63
64 /*Cost: 8M + 8a */
65 inline static void ec_fp_dbl_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1
    , const uni T1){
66     uni_t r0[FP_LEN], r1[FP_LEN], r2[FP_LEN], r3[FP_LEN];
67
68     fp_mul_256(r0, X1, T1);
69     fp_mul_256(r1, Y1, Z1);
70     fp_mul_256(r2, T1, Z1);
71     fp_mul_256(r3, X1, Y1);
72
73     fp_add_256(X3, r0, r1);
74     fp_sub_256(Y3, r0, r1);
75     fp_sub_256(T3, r2, r3);
76     fp_add_256(Z3, r2, r3);
77
78     fp_mul_256(r0, X3, T3);
79     fp_mul_256(r1, Y3, Z3);
80     fp_mul_256(r2, T3, Z3);
81     fp_mul_256(r3, X3, Y3);
82
83     fp_add_256(X3, r0, r1);
84     fp_sub_256(Y3, r0, r1);
85     fp_sub_256(Z3, r2, r3);
86     fp_add_256(T3, r2, r3);
87
88 }
89
90 /*Cost: 8M + 6a. */
91 static void ec_fp_madd_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1,
    const uni T1, const uni X2, const uni Y2){
92

```

```

93 uni_t r0[FP_LEN], r1[FP_LEN], r2[FP_LEN], r3[FP_LEN], r4[FP_LEN], r5[FP_LEN];
94
95 fp_mul_256(r0, X1, X2);//t1=A
96 fp_mul_256(r1, Z1, X2);//t3=C
97 fp_mul_256(r2, Y1, Y2);//t2=B
98 fp_mul_256(r3, T1, Y2);//t4=D
99
100 fp_add_256(r4, Z1, r0);//t5=E
101 fp_add_256(r1, X1, r1);//t6=F
102 fp_add_256(r5, T1, r2);//t1=G
103 fp_add_256(r3, Y1, r3);//t2=H
104
105 fp_sub_256(r0, Z1, r0);//t3=I
106 fp_sub_256(r2, T1, r2);//t4=J
107
108 fp_mul_256(Z3, r0, r5);
109 fp_mul_256(X3, r1, r2);
110 fp_mul_256(T3, r2, r4);
111 fp_mul_256(Y3, r3, r0);
112
113 }
114 /*Cost: 8M + 6a. */
115 static void ec_fp_msub_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1,
const uni T1, const uni X2, const uni Y2){
116
117 uni_t r0[FP_LEN], r1[FP_LEN], r2[FP_LEN], r3[FP_LEN], r4[FP_LEN], r5[FP_LEN];
118
119 fp_mul_256(r0, X1, X2);//t1=A -
120 fp_mul_256(r1, Z1, X2);//t3=C -
121 fp_mul_256(r2, Y1, Y2);//t2=B -
122 fp_mul_256(r3, T1, Y2);//t4=D -
123
124 fp_sub_256(r4, Z1, r0);//t5=E
125 fp_sub_256(r1, X1, r1);//t6=F
126 fp_sub_256(r5, T1, r2);//t1=G
127 fp_sub_256(r3, Y1, r3);//t2=H
128
129 fp_add_256(r0, Z1, r0);//t3=I
130 fp_add_256(r2, T1, r2);//t4=J
131
132 fp_mul_256(Z3, r0, r5);
133 fp_mul_256(X3, r1, r2);
134 fp_mul_256(T3, r2, r4);
135 fp_mul_256(Y3, r3, r0);
136
137 }
138
139 /*Cost: 10M+10a. */
140 static void ec_fp_add_256H_u(uni X3, uni Z3, uni Y3, uni T3, const uni X1, const uni Z1, const uni Y1, const
uni T1, const uni X2, const uni Z2, const uni Y2, const uni T2){
141
142 uni_t t1[FP_LEN], t2[FP_LEN], t3[FP_LEN], t4[FP_LEN], t5[FP_LEN], t6[FP_LEN], t7[FP_LEN], t8[FP_LEN], t9[
FP_LEN], t10[FP_LEN];
143
144 fp_add_256(t1, X2, Z2);
145 fp_add_256(t2, T2, Y2);
146 fp_mul_256(t3, X1, X2);
147 fp_mul_256(t4, Y1, Y2);
148 fp_mul_256(t5, Z1, Z2);
149 fp_mul_256(t6, T1, T2);
150
151 fp_add_256(t7, Z1, X1);
152 fp_add_256(t8, T1, Y1);
153 fp_add_256(t9, t3, t5);
154 fp_add_256(t10, t4, t6);
155
156 fp_mul_256(t1, t7, t1);
157 fp_mul_256(t2, t8, t2);
158
159 fp_sub_256(t8, t6, t4);
160 fp_add_256(t6, t6, t4);
161 fp_sub_256(t7, t5, t3);
162 fp_add_256(t5, t5, t3);
163 fp_sub_256(t1, t1, t9);
164 fp_sub_256(t2, t2, t10);
165
166 fp_mul_256(X3, t1, t8);
167 fp_mul_256(Z3, t7, t6);
168 fp_mul_256(Y3, t2, t7);
169 fp_mul_256(T3, t5, t8);
170
171 }
172
173 inline static void fp_cnt_256_u(int *bc, const uni an, const uni_t al){
174 uni_t w, i, j;
175
176 for(i = al - 1; (an[i] == 0) && (i > 0); i--);
177 w = an[i];
178 for(j = 0; w != 0; j++){
179 w >>= 1;
180 }
181 (*bc) = j + i*UNIT_LEN;
182 }
183
184 inline static void find_nextwindow_u(int *v, int *k, int *wd, const uni en, const int i){
185 int t, u2, s, b, sl, sr;
186 uni_t n;
187

```



```

188 b = i/UNIT_LEN;
189 sr = (i + 1) - (b*UNIT_LEN);
190 sl = UNIT_LEN - sr;
191 if(b < FP_LEN){
192     n = en[b] << sl;
193 }
194 else{
195     n = 0;
196 }
197 if((b != 0) && (sl != 0)){
198     n |= (en[b - 1] >> sr);
199 }
200 t = n >> (UNIT_LEN - 1);
201 if(((n >> (UNIT_LEN - 2)) & 0x1) == t){
202     *v = 0; *k = i; *wd = 1;
203 }
204 else{
205     if(WINDOW_SIZE_LTR < (i + 1)){
206         *wd = WINDOW_SIZE_LTR;
207     }
208     else{
209         *wd = i + 1;
210     }
211     n <<= 1;
212     n >>= (UNIT_LEN - *wd);
213     if((i - *wd + 1) < 1){
214         u2 = 0;
215     }
216     else{
217         u2 = n & 0x1;
218     }
219     *v = -(t << (*wd - 1)) + (n >> 1) + u2;
220     s = 0;
221     for(;; (*v & 0x1) == 0; s++, *v = *v >> 1);
222     *k = i - (*wd - 1) + s + 1;
223 }
224 }
225
226 void ec_fp_smul_256h_u(uni X1, uni Z1, uni Y1, uni T1, const uni kn, const uni X2, const uni Y2)
227 {
228     uni_t X[TABLE_SIZE_LTR][FP_LEN], Y[TABLE_SIZE_LTR][FP_LEN], Z[TABLE_SIZE_LTR][FP_LEN], T[TABLE_SIZE_LTR][
229     FP_LEN];
230     int i, j, ni, k, wd;
231     fp_cnt_256_u(&i, kn, FP_LEN);
232     if(i == 0){
233         fp_set_1_256(X1, 0);
234         fp_set_1_256(Z1, 1);
235         fp_set_1_256(Y1, 0);
236         fp_set_1_256(T1, 1);
237     }
238     else{
239         ec_fp_mdbl_256H_u(X[0], Z[0], Y[0], T[0], X2, Y2); /* 2P. */
240         ec_fp_madd_256H_u(X[1], Z[1], Y[1], T[1], X[0], Z[0], Y[0], T[0], X2, Y2); /* 3P. */
241         for(j = 2; j < TABLE_SIZE_LTR; j++){
242             ec_fp_add_256H_u(X[j], Z[j], Y[j], T[j], X[j - 1], Z[j - 1], Y[j - 1], T[j - 1], X[0], Z[0], Y[0], T
243             [0]);
244         }
245         find_nextwindow_u(&ni, &k, &wd, kn, i);
246         i -= wd;
247         if(ni > 0){
248             ni >>= 1;
249             if(ni == 0){
250                 ec_fp_cpy_256H_u(X1, Z1, Y1, T1, X2, NULL, Y2, NULL);
251             }
252             else{
253                 ec_fp_cpy_256H_u(X1, Z1, Y1, T1, X[ni], Z[ni], Y[ni], T[ni]);
254             }
255         }
256         else{
257             ni = (-ni) >> 1;
258             if(ni == 0){
259                 ec_fp_neg_256H_u(X1, Z1, Y1, T1, X2, NULL, Y2, NULL);
260             }
261             else{
262                 ec_fp_neg_256H_u(X1, Z1, Y1, T1, X[ni], Z[ni], Y[ni], T[ni]);
263             }
264         }
265         for(j = k - 1; i >= 0; j--){
266             find_nextwindow_u(&ni, &k, &wd, kn, i);
267             i -= wd;
268             if(ni == 0){
269                 ec_fp_dbl_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1); //DBL1++;
270             }
271             else{
272                 for(; j > k; j--){
273                     ec_fp_dbl_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1); //DBL2++;
274                 }
275                 ec_fp_dbl_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1); //DBL2++;
276             }
277             if(ni > 0){
278                 ni >>= 1;
279                 if(ni == 0){
280                     ec_fp_madd_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1, X2, Y2); //DBL3++; ADD++;
281                 }
282                 else{
283                     ec_fp_add_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1, X[ni], Z[ni], Y[ni], T[ni]); //DBL3++; ADD++;
284                 }
285             }
286         }
287     }
288 }

```

```

284     else{
285         ni = (-ni) >> 1;
286         if(ni == 0){
287             ec_fp_msub_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1, X2, Y2); //ADD++;
288         }
289         else{
290             ec_fp_add_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1, X[ni], Z[ni], Y[ni], T[ni]); //ADD++;
291         }
292     }
293 }
294 }
295 for(; j >= 1; j--){
296     ec_fp_add_256H_u(X1, Z1, Y1, T1, X1, Z1, Y1, T1, X1, Z1, Y1, T1);
297 };
298 }
299 }
300 }

```

algorithm 6.2: Codes/p1Huff/ec_fp_256h_u.c

