CERIAS Tech Report 2009-07

Low Genus Algebraic Curves in Cryptography

by Ning Shang

Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ning Shang

Entitled Low Genus Algebraic Curves in Cryptography

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Samuel Wagstaff, Jr. Chair	Freydoon Shahidi
Tzuong-Tsieng Moh	Edray Goins
Michael Jacobson, Jr.	

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Samuel Wagstaff, Jr.

Michael Jacobson, Jr.

Approved by: Steven Bell 1/23/2009
Head of the Graduate Program Date

Graduate School Form 20
(Revised 10/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
Low Genus Algebraic Curves in Cryptography

For the degree of Doctor of Philosophy

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Ning Shang
Signature of Candidate

3/12/2009
Date

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

LOW GENUS ALGEBRAIC CURVES IN CRYPTOGRAPHY

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ning Shang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2009

Purdue University

West Lafayette, Indiana

To my family.

ACKNOWLEDGMENTS

I would like to take the opportunity to express my thanks to all the people who have helped, supported and encouraged me in these years. Without you, this thesis work would have been Mission Impossible.

My deepest gratitude goes to my advisor, Dr. Samuel Wagstaff, Jr., and my co-advisor, Dr. Michael Jacobson, Jr., for their invaluable support and guidance.

I thank all my thesis committee members, Dr. Edray Goins, Dr. Tzuong-Tsieng Moh and Dr. Freydoon Shahidi. I appreciate their guidance, encouragement, and providing very helpful comments to this thesis.

I am deeply grateful to Dr. Elisa Bertino for her enormous support and encouragement. It is my honor to work as a member of her research team.

I appreciate the hospitality I received from Microsoft Corportation during my internship in summer 2007. I wish to express my warmest thanks to my mentor, Dr. Kristin Lauter. I learned a great deal from working with her. Her knowledge, spirit, and positive attitude always stimulate me to explore new areas and to work harder. I also very much enjoyed the soccer games we played together.

I would like to convey my thanks to Dr. Joseph Lipman, Dr. Jiu-Kang Yu, and all other wonderful professors in mathematics, computer science, and electrical engineering. I have broadened my knowledge by receiving the various valuable lectures from them. I benefited greatly from useful discussions with them, and am grateful for their patience and kindness.

The 2006 summer school in Wyoming was an unforgettable experience. I am grateful to Dr. Renate Scheidler, Dr. Andreas Stein for their supervision on the research project, and to Dr. Stefan Erickson, Dr. Shuo Shen, and other team members for helpful meetings and discussions. Hiking in snow-covered mountain trails under the beautiful summer sky of Wyoming was definitely among the best parts of life.

To my dearest family and friends, I owe too much love. Their contant support, care, and friendship have accompanied me through many hard times and joyful moments.

Finally, I wish to thank Mary Gitzen, who helped me improve my English when I first came to the States, Dr. Shelborne, who magically fixed my broken knee, and all the nice people I met at Purdue. A thesis is a result of years of hard work. Their help is priceless!

TABLE OF CONTENTS

Page

ABSTRACT vii

1 INTRODUCTION 1
  1.1 Background 1
  1.2 Contribution of the Thesis 6
  1.3 Structure of the Thesis 7

2 A CRYPTOGRAPHIC APPLICATION OF ELLIPTIC CURVES 9
  2.1 Background 9
  2.2 Definitions and Notation 10
  2.3 Key Management Scheme Using Elliptic Curves 12
    2.3.1 Initialization 12
    2.3.2 Encrypting Key Generation 13
    2.3.3 User Subscription 14
    2.3.4 Decrypting Key Derivation 15
    2.3.5 An Example 16
  2.4 Analysis of The Scheme 17
    2.4.1 Tamper-resistant Devices 17
    2.4.2 Hash Functions and ECDLP 18
    2.4.3 Security Against Possible Attacks 18
    2.4.4 Yet Another Good Feature 23
    2.4.5 Space and Time Complexity 23
  2.5 Future Work and Remarks 25

3 ARITHMETIC ON JACOBIANS OF GENUS 2 REAL HYPERELLIPTIC CURVES 27
  3.1 Introduction and Motivation 27
  3.2 Background 28
  3.3 Equivalent Change of Coordinates 29
    3.3.1 Elliptic Curves 30
    3.3.2 Imaginary Hyperelliptic Curves 30
    3.3.3 Real Hyperelliptic Curves 31
  3.4 Explicit Formulas for Elliptic and Genus 2 Imaginary Hyperelliptic Curves 34
    3.4.1 Mumford Representation and Cantor's algorithm 34
    3.4.2 Elliptic Curves And Genus 2 Hyperelliptic Curves in the Imaginary Model 35

Page
3.5 Genus 2 Hyperelliptic Curves in Real Model 39
3.6 Explicit Formulas for the Real Model 41
    3.6.1 Baby Step and Addition Formulas 44
    3.6.2 Doubling Formulas 44
    3.6.3 Summary of Results 57
3.7 Future Work 59

4 GENERATING SUITABLE PARAMETERS FOR DISCRETE-LOG BASED CRYPTOGRAPHY WITH POLYNOMIAL PARAMETERIZATION . . 61
4.1 Introduction 61
4.2 Algorithms 62
4.3 Probability That $p(x)$ is Prime and $N_i(x)$ is Almost Prime 69
4.4 Conclusion and Further Discussion 73

5 GENERATING PARAMETERS FOR PAIRING-FRIENDLY GENUS 2 CURVES OVER PRIME FIELDS 75
5.1 Introduction 75
5.2 Weil and Tate-Lichtenbaum Pairings 76
5.3 Pairing-friendly Genus 2 Curves Are Rare 77
5.4 Algorithms for Generating Pairing-friendly Genus 2 Curves over Prime Fields 82
5.5 Generating Pairing Parameters with Polynomial Parameterization . 85
5.6 Updates on Related Research and Future Work 89

6 CONCLUSIONS AND FUTURE WORK 91

LIST OF REFERENCES 93

A Parameters for Discrete Log Based Cryptography 99

B Pairing-friendly Genus 2 Curves: Numerical Data 103
B.1 Parameters produced by Algorithm 7 103
B.2 Parameters produced by Algorithm 8 109

C Some Source Code 117
C.1 PARI/GP scripts for finding parameters for cryptographically-strong genus 2 curves 117
C.2 MAGMA scripts for finding parameters for pairing-friendly genus 2 curves 122

VITA 125

ABSTRACT

Ning Shang Ph.D., Purdue University, May, 2009. Low Genus Algebraic Curves in Cryptography. Major Professors: Samuel S. Wagstaff, Jr. and Michael J. Jacobson, Jr.

Preserving a strong connection between mathematics and information security, elliptic and hyperelliptic curve cryptography are playing an increasingly important role during the past decade. We present some problems that relate low genus curves and cryptography.

We first discuss a new application of elliptic curve cryptography (ECC) to a real-world problem of access control in secure broadcasting of data. The asymmetry, introduced by the elliptic curve discrete logarithm problem, is the key to achieving the required security feature that existing methods fail to obtain.

We then talk about the use of genus 2 curves in the "real model" in cryptography, and present explicit divisor doubling formulas for such curves. These formulas are particularly important for implementation purposes.

Finally, we present a new method for finding cryptographically strong parameters for the CM construction of genus 2 curves. This method uses the idea of polynomial parameterization, which allows suitable parameters to be generated in batches. We give a brief analysis of the algorithm. We also provide algorithms for generating parameters for genus 2 curves to be used in pairing-based cryptography. Our method is an adaptation of the Cocks-Pinch construction for pairing-friendly elliptic curves. Our methods start from a prescribed embedding degree k and a primitive quartic CM field K, and output a prime subgroup order r of the Jacobian over a prime field $\mathbb{F}_p$, with $\rho = 2\log(p)/\log(r) \approx 8$.

1. INTRODUCTION

1.1 Background

Information security is playing an increasingly important role as communications over computer networks and the deployment of digital storage media start to spread their domination over the world. *Cryptography*, as the "study of mathematics techniques related to aspects of information security such as confidentiality, data integrity, entity identification, and data origin authentication" [1], since its first invention in the ancient times, has seen a remarkably rapid development in recent decades.

The 1976 paper [2] of W. Diffie and M. Hellman brought to people's attention for the first time one of the most important discoveries in the history of cryptography, the notion of *public-key cryptography.* Based on the idea of *trapdoor functions*, public-key cryptography (aka asymmetric cryptography) has provided practical new solutions to many problems in information security, such as secure key exchange over non-secure channels, authentication and digital signatures, which traditional secret-key (aka symmetric-key) cryptography alone is unable to do. Though advantageous to use for secure communications, public-key cryptography is computationally costly for encryption/decryption compared to secret-key cryptographic algorithms. Hence in many cases, public-key cryptography is used to transmit the symmetric keys of secret-key algorithms. In the meantime, a lot of effort has been put into research of efficiency improvement for public-key cryptographic schemes.

Nowadays, two types of public-key cryptosystem have survived the examination of researchers and practitioners in areas of academia and industry, and are regarded as practical to use. Among these two types, schemes like RSA [3] were first recognized and studied; these are based on the difficulty of factoring large integers.

The other type of system is based on the discrete logarithm problem (DLP) in certain finite cyclic groups. The Pohlig-Hellman algorithm [4] for solving the DLP implies that only finite groups of prime order are suitable candidates. Such a system can be implemented in various ways, e.g., by using the multiplicative group of the invertible elements of a finite field [2] or abelian varieties over finite fields [5, 6]. The best known generic algorithms such as Shanks' Baby Step Giant Step, Pollard's ρ and λ methods solve the DLP in exponential time $O(\sqrt{N})$, where N is the order of the cyclic group. In every efficient implementation, besides the intractability of the DLP, two related fundamental questions need to be addressed: 1) compact representation (i.e. encoding) of group elements and 2) efficient arithmetic for the group operation. As abelian varieties, elliptic curves succeed in providing positive answers to both the above questions. Because no subexponential algorithm is known for solving the elliptic curve discrete logarithm problem (ECDLP) in general, they outdo the multiplicative groups $\mathbb{F}_q^\times$ of finite fields $\mathbb{F}_q$ in offering the same level of security with a faster speed. Note that there exist subexponential methods (e.g. the index calculus, see [7]) for solving the DLP in $\mathbb{F}_q^\times$.

The theory of elliptic curves has been studied extensively during past centuries. Their application to cryptography helps promote further research. Motivated by the case of elliptic curves, cryptographic research on Jacobians of curves of higher genus has started to emerge and attract attention. The use of Jacobians of curves of higher genus has the advantage of being suitable for implementation on small processor architectures. However, for hyperelliptic curves of genus ≥ 3 there are index calculus attacks (see [8–13]) which are faster than the generic attacks. This implies a potential insecurity of using such hyperelliptic curves from a cryptographic perspective[1]. In this thesis, we discuss problems related to elliptic curves (genus 1) and hyperelliptic curves of genus 2.

---

[1] The best attack described in [13] takes time $\tilde{O}(q^{2-\frac{2}{g}})$, which is asymptotically slightly faster than the generic algorithms. Here $\tilde{O}(f(q))$ is a function that is bounded by $f(q)$ times a polynomial in $\log(f(q))$ for large enough q.

We define an algebraic curve C to be a projective variety of dimension one. Moreover, in this thesis, we always work on a nonsingular affine model of the curve defined over the underlying perfect field K of interest, which usually is a finite field $\mathbb{F}_q$. Furthermore, we assume the curve is *absolutely irreducible*, i.e., it is irreducible as a closed set with respect to the Zariski topology of projective space $\mathbb{P}^2$ over $\bar{K}$, a separable (algebraic) closure of K.

We define an elliptic curve E over K to be a nonsingular absolutely irreducible algebraic curve defined over K of genus 1 with one K-rational point ∞, and refer to it by its model given by the following Weierstraß equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in K.$$

We define a hyperelliptic curve C over K to be a nonsingular absolutely irreducible algebraic curve of genus $g \geq 2$, given by a model

$$C : y^2 + h(x)y = f(x), \quad h(x), f(x) \in K[x], \deg(h) \leq g+1, \deg(f) \leq 2g+2,$$

such that the associated function field $K(C)$, which is the field of fractions of the coordinate ring of C, is a separable extension of degree 2 of the rational function field $K(x)$ of $\mathbb{P}^1_K$ for some function x; We will introduce another definition of a hyperelliptic curve and discuss more about it in Chapter 3.

Given an algebraic curve C over a field K, a *divisor* of C over K is a formal sum

$$D = \sum_{\mathfrak{p}} n_{\mathfrak{p}}\mathfrak{p},$$

where $\mathfrak{p}$ runs over all places of the function field $K(C)$, $n_{\mathfrak{p}} \in \mathbb{Z}$, and only finitely many $n_{\mathfrak{p}}$ are different from 0. The degree of the residue field $K(C)_{\mathfrak{p}}/K$ is defined to be $\deg(\mathfrak{p})$; the *degree* of a divisor D is $\sum_{\mathfrak{p}} n_{\mathfrak{p}} \deg(\mathfrak{p})$. A divisor D is called *effective*, written $D \geq 0$, if we have all $n_{\mathfrak{p}} \geq 0$. For any function $f \in K(C)^*$ we define the *divisor of* f to be

$$(f) = \sum_{\mathfrak{p}} v_{\mathfrak{p}}(f)\mathfrak{p},$$

where $v_{\mathfrak{p}}$ is the usual discrete valuation defined using uniformizers at $\mathfrak{p}$. We also call a divisor obtained in this way a *principal divisor*. Note that (f) is a divisor of degree 0. We say two divisors D_1 and D_2 are *equivalent*, written $D_1 \sim D_2$ if $D_1 = D_2 + (f)$ for some $f \in K(C)$.

Definition 1 (Riemann-Roch Space of a Divisor) *For a divisor D of an algebraic curve C over K, the Riemann-Roch space of D is*

$$L(D) = \{f \in K(C)^* | (f) + D \geq 0\} \cup \{0\}.$$

It follows easily that $L(D)$ is a vector space over K, and $L(D_1) \simeq L(D_2)$ as K-vector spaces if $D_1 \sim D_2$. We write $\dim L(D)$ as the K-dimension of $L(D)$.

Theorem 2 (Riemann-Roch [14]) *Let C be an absolutely irreducible algebraic curve over K with function field $K(C)$. There exists an integer $g \geq 0$ such that for every divisor D of C over K*

$$\dim L(D) \geq \deg(D) - g + 1.$$

For all divisors D with $\deg(D) > 2(g-1)$*, one has equality*

$$\dim L(D) = \deg(D) - g + 1.$$

Definition 3 (Genus [14]) *The integer g from Theorem 2 is called the genus of $K(C)$ or the geometric genus of C. If C is projective nonsingular then g is called the genus of C.*

Note that if C has genus 1, then it is elliptic; if C has genus 2, the it is hyperelliptic.

Let $Div_K^0(C)$ be the set of all degree 0 divisors defined over K, i.e., stable under the Galois action of $Gal(\bar{K}/K)$. Let $Prin_K(C)$ be the set of all principal divisors (associated with function $f \in K(C)^*$). Then $Prin_K(C)$ is a subgroup of $Div_K^0(C)$. Note that $Prin_K(C)$ is the image of the map

$$\phi : \quad K(C)^* \to Div_K^0(C), \quad f \mapsto (f).$$

Definition 4 (Divisor Class Group) *The divisor class group of an algebraic curve C over a field K is the quotient group*

$$Pic_K^0(C) = Div_K^0(C)/Prin_K(C).$$

From an arithmetic point of view, we define the *Jacobian variety*, or *Jacobian*, of a curve C to be the divisor class group of C over $\bar{K}$, an algebraic closure of K, i.e.,

$$Jac_{\bar{K}}(C) = Pic_{\bar{K}}^0(C).$$

This is equivalent to saying that the following sequence is short exact:

$$\{1\} \to \bar{K}(C)^*/\bar{K}^* \xrightarrow{\phi} Div_{\bar{K}}^0(C) \to Jac_{\bar{K}}(C) \to \{0\}, \tag{1.1}$$

where $\phi : \bar{K}(C)^*/\bar{K}^* \to Div_{\bar{K}}^0(C)$ is given by $\phi(f) = (f)$.

Let $Gal(\bar{K}/K)$ be the absolute Galois group of K. $\forall \sigma \in Gak(\bar{K}/K)$, σ induces an action on $C_{\bar{K}}$ by

$$\sigma : (x_0, y_0) \mapsto (\sigma(x_0), \sigma(y_0)),$$

where $\mathfrak{p} \in C(\bar{K}) = (x_0, y_0)$ is a point in $C_{\bar{K}}$, hence it induces an action on $Jac_{\bar{K}}(C)$ by

$$\sigma : D \mapsto \sum n_{\mathfrak{p}} \sigma(\mathfrak{p}),$$

where $D = \sum n_{\mathfrak{p}} \mathfrak{p}$. We define the *Jacobian over K of C*, or the *K-rational points of the Jacobian* to be

$$Jac_K(C) = Jac_{\bar{K}}(C)^{Gal(\bar{K}/K)},$$

i.e., the elements fixed by the action of $Gal(\bar{K}/K)$.

Taking Galois cohomology of (1.1), we obtain

$$1 \to K(C)^*/K^* \to Div_K^0(C) \to Jac_K(C) \to H^1(Gal(\bar{K}/K), Prin_{\bar{K}}(C)), \tag{1.2}$$

where $Gal(\bar{K}/K)$ acts canonically on $Prin_{\bar{K}}(C)$.

Therefore, the divisor class group of C over K is a subgroup of the K-rational points of the Jacobian. If furthermore, the curve C has a K-rational point, then the Jacobian of a curve over any field K can be identified with its divisor class group over K, by the following theorem.

Theorem 5 (Galbraith, et al., 1998, [15]) *Let C/K be a curve with a K-rational point. Then*

$$H^1(Gal(\bar{K}/K), Prin_{\bar{K}}(C)) = \{0\}.$$

In this thesis, this is always the case we consider. We shall identify $Jac_K(C)$ with $Pic^0_K(C)$ throughout the discussion.

1.2 Contribution of the Thesis

The research described in this thesis focuses on solving mathematical problems related to elliptic and hyperelliptic curve cryptography. The thesis makes three main contributions to the field of elliptic and hyperelliptic curve cryptography.

First, the thesis proposes a encryption/decryption key management scheme for access control in a hierarchy for which the keys are updated with time. Such a scheme has a practical important use in settings like communications and e-commerce. The method proposed in the thesis employs elliptic curve cryptography in construction of the key management scheme, which makes it resistant to attacks that break earlier proposals of such schemes. The key management scheme is published in [16]. The portion of the work described in Chapter 2 of this thesis was done entirely by the author at the suggestion of Professors E. Bertino and S. Wagstaff.

Schemes like the above can be alternatively implemented by using genus 2 curves. The arithmetic on such curves needs to be considered. Second, the thesis gives explicit formulas for divisor doubling for “real” genus 2 hyperelliptic curves over finite fields of positive characteristic. Such explicit formulas are useful for efficient implementation of cryptographic protocols (see, e.g. [17]) using the infrastructure of the principal ideal class of the function field associated with the curve. The formulas presented in this thesis cover the most common case in divisor doubling arithmetic as well as all special cases, for two major representations of divisors. This consists of part of a research project the author participated in with Dr. S. Erickson, Professor M. Jacobson, Dr. S. Shen and Professor A. Stein. Other formulas and improvements are being developed by other researchers. Theorem 8 for equivalent change of coordinates of real hyperelliptic curves in Section 3.3.3 was suggested and proved by the author. The divisor class doubling formulas, including the most common case and special cases, presented in Section 3.6.2 were derived by the author.

In order to use Jacobians of genus 2 hyperelliptic curves for discrete logarithm based or pairing based cryptography, some parameters, e.g. the underlying finite field, the cardinality of the Jacobian, and the "embedding degree," need to be considered. The third contribution of the thesis includes new approaches to generating such parameters for the complex multiplication method of constructing equations of genus 2 curves. The contribution also includes analysis of the polynomial parameterization method of generating cryptographically strong parameters for genus 2 curves, following the Bateman-Horn philosophy, and a quantitative analysis of the scarcity of "pairing-friendly" genus 2 curves, based on the Riemann Hypothesis. This research was suggested by Dr. K. Lauter during the author's internship at Microsoft Research in 2007. The portion of the work described in Chapters 4 and 5 was entirely done by the author.

1.3 Structure of the Thesis

The rest of the thesis is organized as follows: Chapter 2 describes a solution to a practical problem of access control for secure broadcasting of information which uses the elliptic curve discrete logarithm problem. Chapter 3 presents some results on the arithmetic of genus 2 real hyperelliptic curves, which is useful for a class of DLP-like cryptographic protocols. Chapter 4 shows a method to generate parameters for constructing genus 2 curves via the complex multiplication (CM) method. Chapter 5 reports progress on finding parameters for generating pairing-friendly genus 2 curves over prime fields for the CM method. Chapter 6 summarizes and suggests future work.

2. A CRYPTOGRAPHIC APPLICATION OF ELLIPTIC CURVES

In this chapter we present an efficient time-bound hierarchical key management scheme for secure broadcasting of encrypted data content. Elliptic curve cryptography is used in the construction of the scheme to help resist attacks that break earlier schemes. Part of the research described in this chapter can be found in [16].

2.1 Background

In a web-based environment, such as one involving electronic newspapers, data can be organized according to different access control policies, encrypted using distinct encryption keys, then broadcast to all users. Usually these data can be organized as a hierarchical tree. We need a key management scheme so that a higher class can retrieve information that a lower class is authorized to access — but not the other way around. Moreover, in many applications, such as electronic newspaper/journal subscription, pay TV broadcasting, etc., there is a time bound associated with each access control policy, so that a user is assigned to a certain class for just a period of time. The users' keys need to be updated periodically to ensure that the delivery of the information follows the access control policies of the data source. An ideal time-bound hierarchical key management scheme should be able to perform the above task in an efficient manner and minimize the storage and communication of keys. In 2002, W.G. Tzeng [18] attempted to solve this problem. W.G. Tzeng's scheme is efficient in terms of its space requirement, but is computationally inefficient, since a Lucas function operation is used to construct the scheme, and this incurs heavy computational load. Moreover, it is insecure against collusion attacks as shown by X. Yi and Y. Ye [19].

Another time-bound hierarchical key assignment scheme, based on a tamper-resistant device and a secure hash function, was proposed by H.Y. Chien [20] in 2004. This scheme greatly reduces computational load and implementation cost. However, it has a security hole against X. Yi 's three-party collusion attack [21]. Inspired by H.Y. Chien's idea, we propose in this thesis a new method for access control using elliptic curve cryptography. This scheme is efficient and secure against X. Yi 's three-party collusion attack.

Although there have been attacks on smart cards [22] and some other tamper-resistant devices, such attacks require special equipment which would cost more than a subscription. The only really valuable data on the smart cards our scheme uses is the master key. It must be kept secret because an attacker who obtained it could derive all the keys for the data that one could get with this smart card. Assuming the master key can be protected, there is good reason to believe that our scheme that uses tamper-resistant devices can have practical important applications, in areas such as digital rights management.

Our original motivation for this work was to provide a better key management scheme for [23], in which data are encoded in XML and broadcast to a hierarchy.

The rest of this chapter is organized as follows: Section 2.2 presents the notation and definitions needed to give a hierarchical structure to the data source. Section 2.3 proposes the new time-bound key management scheme applied to a hierarchy. Section 2.4 contains further discussion of the key management scheme.

2.2 Definitions and Notation

Let $\mathcal{S}$ be the data source to broadcast. We assume $\mathcal{S}$ is partitioned into blocks of data called *nodes*.

The policy base $\mathcal{PB}$ is the set of access control policies defined for $\mathcal{S}$. In our setting, each access control policy $\mathsf{acp} \in \mathcal{PB}$ contains a temporal interval I among its components, which specifies the time period in which the access control policy is valid. A sample access control policy for XML documents might look like

$$\mathsf{acp} = (\mathsf{I}, \mathsf{P}, \mathsf{sbj\text{-}spec}, \mathsf{prot\text{-}obj\text{-}spec}, \mathsf{priv}, \mathsf{prop\text{-}opt}),$$

where I, P, $\mathsf{sbj\text{-}spec}$, $\mathsf{prot\text{-}obj\text{-}spec}$, priv and $\mathsf{prop\text{-}opt}$ are the temporal interval, the periodic expression, the credential specification, the protection object specification, the privilege and the propagation option of acp, respectively. For example, the temporal interval I may specify a time period in which a particular resource can be used by a particular entitiy. Since this chapter does not focus on access control policies, we do not intend to get into more details about them. Interested readers may refer to [24] and [23] for details.

It is important to notice that several policies may apply to each node in $\mathcal{S}$. In what follows we refer to the set of policies applying to a node in $\mathcal{S}$ as the **policy configuration** associated with the node. Also, in what follows $\mathcal{PC}_{\mathcal{PB}}$ denotes the set of all possible policy configurations which can be generated by policies in $\mathcal{PB}$.

We now introduce the notion of a class of nodes, a relevant notion in our approach. Intuitively, a class of nodes corresponds to a given policy configuration and identifies all nodes to which the configuration applies. Intuitively, a class of nodes includes the set of nodes to which the same set of access control policies apply.

Definition 1 (Class of nodes) *Let Pc_i be a policy configuration belonging to $\mathcal{PC}_{\mathcal{PB}}$. The* **class of nodes marked with** *Pc_i, denoted by $\mathcal{C}_i$, is the set of nodes belonging to the data source $\mathcal{S}$ marked by all and only the policies in Pc_i. Note that the empty set could be a class of nodes marked with a certain policy configuration. We denote by $\mathcal{C}$ the set of all classes of nodes defined over $\mathcal{S}$ marked with the policy configurations in $\mathcal{PC}_{\mathcal{PB}}$, and we have the following requirement: we distinguish and include in $\mathcal{C}$ the empty sets, if marked by policy configurations consisting of only one access control policy, and exclude from $\mathcal{C}$ the empty sets marked by any other policy configurations. Note that $\mathcal{C}$ corresponds to a subset of $\mathcal{PC}_{\mathcal{PB}}$.*

We distinguish and include the empty sets corresponding to different singleton policy configurations so that keys can be assigned to these classes, which enable users belonging to these classes to derive required decryption keys of lower classes. This key derivation process will be described in Section 2.3.

The idea for the secure broadcasting mode of the data source is this: the portions of the source marked by different classes of nodes are encrypted by different secret keys, and are broadcast periodically to the subscribers. Subscribers receive only the keys for the document sources that they can access according to the policies.

The following definition introduces a partial order relation defined over $\mathcal{C}$.

Definition 2 (Partial order relation on $\mathcal{C}$) *Let $\mathcal{C}_i$ and $\mathcal{C}_j$ be two classes of nodes marked by Pc_i and Pc_j, respectively, where Pc_i and Pc_j are policy configurations in $\mathcal{PC}_{\mathcal{PB}}$. We say that $\mathcal{C}_i$ dominates $\mathcal{C}_j$, written $\mathcal{C}_j \preceq \mathcal{C}_i$, if and only if $Pc_i \subseteq Pc_j$. We also write $\mathcal{C}_j \prec \mathcal{C}_i$ if $\mathcal{C}_j \preceq \mathcal{C}_i$ but $\mathcal{C}_j \neq \mathcal{C}_i$. We also say that $\mathcal{C}_i$ directly dominates $\mathcal{C}_j$, written $\mathcal{C}_j \prec_d \mathcal{C}_i$, if and only if $\mathcal{C}_i \neq \mathcal{C}_j$ and $\mathcal{C}_j \preceq \mathcal{C}_* \preceq \mathcal{C}_i$ implies $\mathcal{C}_* = \mathcal{C}_i$ or $\mathcal{C}_* = \mathcal{C}_j$. We call "$\mathcal{C}_j \prec_d \mathcal{C}_i$" a directed edge. We say $\mathcal{C}_i$ dominates $\mathcal{C}_j$ via n directed edges if there exists $\{\mathcal{C}_{i_k}\}_{1 \leq k \leq n-1} \subseteq \mathcal{C}$ such that $\mathcal{C}_j \prec_d \mathcal{C}_{i_1}$, $\mathcal{C}_{i_{n-1}} \prec_d \mathcal{C}_j$ and $\mathcal{C}_{i_{k-1}} \prec_d \mathcal{C}_{i_k}$ for $2 \leq k \leq n-1$.*

2.3 Key Management Scheme Using Elliptic Curves

2.3.1 Initialization

Suppose we have already generated the set $\mathcal{C}$ of classes of nodes of the data source $\mathcal{S}$ marked with the policy configurations Pc_i in $\mathcal{PB}$. Such a set is partially ordered with respect to $\preceq$. Let n be the cardinality of $\mathcal{C}$.

In this step, the system parameters are initialized and the system's class keys K_i are generated.

1. The vendor chooses an elliptic curve E over a finite field $\mathbb{F}_q$ so that the discrete logarithm problem is hard on $E(\mathbb{F}_q)$.[1] The vendor also chooses a point $Q \in E(\mathbb{F}_q)$ with a large prime order, say, p. The vendor then chooses $2n$ integers n_i, g_i, relatively prime to p, such that $n_i g_i$ are all different modulo p for $1 \leq i \leq n$. The vendor computes $P_i = [n_i]Q$ on $E(\mathbb{F}_q)$ and h_i such that $g_i h_i \equiv 1 \pmod{p}$. The class key $K_i = [g_i]P_i$ is computed for class $\mathcal{C}_i$. The points $R_{i,j} = g_i K_j + ([-1]K_i)$ are also computed whenever $\mathcal{C}_j \prec \mathcal{C}_i$ (not just when $\mathcal{C}_j \prec_d \mathcal{C}_i$).

2. The vendor chooses two random integers a, b and a keyed-hash message authentication code (HMAC) [26] $H_K(-)$ built with a hash function $H(-)$ and a fixed secret key K. K serves as the system's master key and is only known to the vendor.

3. The vendor publishes $R_{i,j}$ on an authenticated board, whereas the integers g_i, h_i, a and b are kept secret. Parties can verify the validity of the $R_{i,j}$ obtained from the board. This can be realized by using digital signatures.

The public values $R_{i,j}$ are constructed in such a way that the owner of the key K_j of the lower class $\mathcal{C}_j$ cannot obtain any information about the class key K_i of the higher class $\mathcal{C}_i$ without knowing the secret value g_i, and the owner of the higher class key K_i cannot compute K_j on its own, due to the the difficulty of solving the discrete logarithm problem. It turns out that such construction is secure against the attack [21] which breaks H.Y. Chien's earlier scheme [20]. We will discuss this in section 2.4.3.3.

2.3.2 Encrypting Key Generation

In this step we generate the temporal encryption class keys $K_{i,t}$ at time granule t by using the system's class keys K_i.

---

[1] For more background on elliptic curve cryptography, see [25].

The class of nodes $\mathcal{C}_i \in \mathcal{C}$ is encrypted by a symmetric encryption algorithm, e.g., AES [27]. We denote by $K_{i,t}$ the secret key for $\mathcal{C}_i$ at time granule $t \in [T_b, T_e] = [1, Z]$. The generation process for $K_{i,t}$ is given by the formula below:

$$K_{i,t} = H_K \left(K_i \parallel H^t(a) \parallel H^{Z-t}(b) \parallel ID_i \right),$$

where $H^m(x)$ is the m-fold iteration of $H(-)$ applied to x, ID_i is the identity of $\mathcal{C}_i$, all components of the input of $H(-)$ are encoded as bit strings, and $\parallel$ is the bit string concatenation. Note that we can choose $H(-)$ properly in the initialization process so that the output of H_K is the right length for a key for the symmetric encryption algorithm we use.

The one-way property of the hash function H ensures that $H^t(a)$ and $H^{Z-t}(b)$ can be calculated only when the values $H^{t_1}(a)$ and $H^{Z-t_2}(b)$ are available for some t_1, t_2 with $t_1 \leq t \leq t_2$. This is the idea for the construction of the "time-bound" of the key management scheme.

2.3.3 User Subscription

This is the user subscription phase, in which a tamper-resistant device storing important information is issued to the subscriber.

Upon receiving a subscription request, an appropriate access control policy acp_i is searched until there is a match, then the policy configuration in $\mathcal{PB}$ which contains **only** acp_i is found, and thus the corresponding class of nodes marked with it, say $\mathcal{C}_i$, is identified. Note that $\mathcal{C}_i$, which could be an empty set, is always in $\mathcal{C}$ by the construction in Definition 1. We define the **encryption information**, EncInf_i, as follows:

$$\mathit{EncInf}_i = \{ \left(H^{t_1}(a), H^{Z-t_2}(b) \right) \},$$

where the set on the right side is defined for all acceptable time intervals $[t_1, t_2]$ for acp_i.

The vendor distributes the class key K_i to the subscriber through a secure channel. The vendor also issues the subscriber a tamper-resistant device storing H_K (thus H,

K), E, $\mathbb{F}_q$, ID_i, h_i and $EncInf_i$. There is also a secure clock embedded in the device which keeps track of current time. The device is tamper-resistant in the sense that no one can recover K, h_i, $EncInf_i$, change the values of ID_i, or change the time of the clock.

2.3.4 Decrypting Key Derivation

In this step the temporal keys for a class and the classes below it are reconstructed by the tamper-resistant device.

Assume that the subscription process mentioned above is completed for a subscriber U associated with class $\mathcal{C}_i$. U can then use the information received from the vendor to decrypt the data in class $\mathcal{C}_j$, with $\mathcal{C}_j \preceq \mathcal{C}_i$, as follows:

1. If $\mathcal{C}_j = \mathcal{C}_i$, U inputs only K_i into the tamper-resistant device; otherwise if $\mathcal{C}_j \prec \mathcal{C}_i$, U first retrieves $R_{i,j}$ from the authenticated public board, then inputs it together with the class identity ID_j of $\mathcal{C}_j$ and its secret class key K_i.

2. If K_j is the only input, the next step is executed directly. Otherwise, the tamper-resistant device computes the secret class key of $\mathcal{C}_j$:

$$K_j = [h_i](R_{i,j} + K_i).$$

3. If $t \in [t_1, t_2]$ for some acceptable time interval $[t_1, t_2]$ of acp_i, the tamper-resistant device computes

$$H^t(a) = H^{t-t_1}(H^{t_1}(a)), \qquad H^{Z-t}(b) = H^{t_2-t}(H^{Z-t_2}(b)),$$

   and $K_{j,t} = H_K(K_j \parallel H^t(a) \parallel H^{Z-t}(b) \parallel ID_j)$. Note that the values $H^{t_1}(a)$ and $H^{Z-t_2}(b)$ are pre-computed and stored in the tamper-resistant device.

4. At time granule t, the protected data belonging to class $\mathcal{C}_j$ can be decrypted by applying the key $K_{j,t}$.

2.3.5 An Example

We now provide an example to illustrate the above process.

Consider an electronic newspaper system. Let **one day** be a tick of time in this system and $Z = 70$ be the life time of the system, i.e., the system exists in the temporal interval $[1, 70]$. Let U be a user wishing to subscribe to the sports portion of the newspaper for one week, say, the period $I = [8, 14]$. We could match U with an access control policy $acp_1 =$ ([8,14], All days, Subscriber/type="full", Sports_supplement, view, CASCADE). Then we can find the class of nodes $\mathcal{C}_1$ marked with policy configuration acp_1 from a pre-generated table. These nodes are encrypted and broadcast periodically. U can derive the decryption key for the subscription period using the issued class key K_1 and the tamper-resistant device storing H_K, E, $\mathbb{F}_q$, ID_1, h_1 and $H^8(a)$, $H^{56}(b) = H^{70-14}(b)$. For example, U inputs K_1 into the device. To obtain the decryption key $K_{1,10}$ at time granule $t = 10$, the device computes

$$H^{10}(a) = H^2(H^8(a)), H^{60}(b) = H^4(H^{56}(b))$$

then $K_{1,10} = H_K(K_1 \parallel H^{10}(a) \parallel H^{60}(b) \parallel ID_1)$, the very thing needed. To obtain the decryption key at $t = 13$ for a class $\mathcal{C}_2 \preceq \mathcal{C}_1$, U inputs K_1, ID_2 and $R_{1,2}$ into the device. The device first computes the class key of $\mathcal{C}_2$

$$K_2 = [h_1](R_{1,2} + K_1).$$

Then it computes

$$H^{13}(a) = H^5(H^8(a)), H^{57}(b) = H(H^{56}(b))$$

and $K_{2,13} = H_K(K_2 \parallel H^{13}(a) \parallel H^{57}(b) \parallel ID_2)$, the decryption key needed.

Note that all computations are executed by the tamper-resistant device. The device can prevent the results of the computations from being revealed, so that even the user U does not know the class key K_2 of the class of nodes $\mathcal{C}_2 \prec \mathcal{C}_1$.

2.4 Analysis of The Scheme

We have proposed a key assignment scheme for secure broadcasting based on a tamper-resistant device. A secure hash function and the intractability of the discrete logarithm problem on elliptic curves over the finite field $\mathbb{F}_q$ are also assumed.

2.4.1 Tamper-resistant Devices

The tamper-resistant device plays an important role in our scheme. The system's master key, K, must be protected by the device. Leak of EncInf_i will not help the attackers much, because they are not able to compute the HMAC, thus the temporal class keys, without knowing K. Although it is unlikely to happen, a leak of h_i will enable the user of class $\mathcal{C}_i$ to obtain the class key K_j of $\mathcal{C}_j$, where $\mathcal{C}_j \preceq \mathcal{C}_i$, by computing

$$K_j = [h_i](R_{i,j} + K_i),$$

as is done by the device. As pointed out by Professor Jacobson in a private correspondence, similarly, a leak of h_k of class $\mathcal{C}_k$ allows the user of class $\mathcal{C}_i \preceq \mathcal{C}_k$ to obtain K_k, by computing

$$K_k = [g_k](K_i + [-h_i]R_{k,i}).$$

Unless K is also discovered, the attacks to retrieve EncInf_i and h_i on individual devices are not effective. With the use of a tamper-resistant device, the security of the scheme is strong enough. From an implementational point of view, the **Trusted Platform Module** (TPM) technology [28], which is good for storing and using secret keys, can well suit our need. We are aware that there are attacks on TPMs [29]. There are countermeasures against those attacks [29]. Moreover, none of these attacks is capable of extracting the exact secret information being protected (in our case, e.g., the system key K). Hence the attackers are not able to perform the HMAC operations. Therefore an attack relying on the knowledge of K is not feasible in practice. We believe the use of the tamper-resistant hardware is practical and secure in reality.

One might argue that if we need such a strong tamper-resistant device, then we might as well store the needed temporal decryption keys on it directly and discard the key management scheme. However, that approach is not practical, because the number of needed keys can be large, considering the temporal intervals and hierarchy. And in that case, the system's class keys can not be easily updated. Our proposed scheme is elegant and more efficient in terms of storage on the tamper-resistant devices.

2.4.2 Hash Functions and ECDLP

Some of the most widely used hash functions, e.g. SHA-0, MD4, Haval-128, RipeMD-128, MD5, were broken years ago; SHA-1 was announced broken early in the year 2005. Essentially, these hash functions have been proven not to be collision-free; but it is still hard to find a pre-image to a given digest in a reasonable time. In view of this, these attacks on hash functions will not affect the security of our scheme, as long as the discrete logarithm problem on the elliptic curves is still hard. So far there is no foreseeable breakthrough in solving DLP on elliptic curves.

Without having to keep $Q \in E(\mathbb{F}_q)$ secret, no one, including the user U_i, can recover the secret values g_i, h_i of the system due to the difficulty of the elliptic curve discrete logarithm problem.

2.4.3 Security Against Possible Attacks

Note that the tamper-resistant device in our scheme is an oracle that does calculation in the Decrypting Key Derivation process. This raises the question of whether such a device can be attacked by an adversary to gain secret information to subvert this process. This concern is necessary since H.Y. Chien's scheme has been successfully attacked (see X. Yi [21]) due to the weakness of the oracle. We face a similar situation here.

We set up the attack model for our scheme as follows:

Attack model

We denote the adversary by $\mathcal{A}$, and assume

1. $\mathcal{A}$ either contains an individual attacker who is not a valid user but captures a device belonging to a user of the system, or a team of valid users who have access to their assigned devices;
2. $\mathcal{A}$ can query the device with trial messages;
3. all the members (if there are multiple ones) of $\mathcal{A}$ share the information and resources they have.

The goal of $\mathcal{A}$ is to derive any valid temporal key $K_{i,t}$ which is not supposed to be used by any member of $\mathcal{A}$.

Based on the attack model above, we shall analyze the security of the proposed scheme. This analysis will not provide proofs of security (i.e. written in the language of provable security), but it will give some ideas how the design of the scheme helps secure the system.

2.4.3.1 Attack From Outside

Suppose an adversary $\mathcal{A}$, who is an individual attacker, captures a device of class $\mathcal{C}_i$, but it does not know the associated class key K_i. $\mathcal{A}$ can query the device with a value K_*, hoping the device to output the valid decryption key $K_{i,t}$ at time t.

We claim that any attempt of $\mathcal{A}$ to gain the temporal decrypting key with only one input K_* to the device with identity ID_i has very low probability of success.

This is so because even if we assume $\mathcal{A}$ queries the device at time granule t which is in the subscription period, we have that the probability that the device outputs the correct decrypting key in response to a randomly chosen query message K_* is

$$\begin{aligned} & \mathit{Prob}\left\{H_K\left(K_* \parallel H^t(a) \parallel H^{Z-t}(b) \parallel \mathit{ID}_i\right) = H_K\left(K_i \parallel H^t(a) \parallel H^{Z-t}(b) \parallel \mathit{ID}_i\right)\right\} \\ = \quad & Pr_1 + Pr_2, \end{aligned}$$

where

$$Pr_1 = Prob\left\{K_* = K_i\right\},$$

and

$$\begin{aligned} Pr_2 = & Prob\, \{K_* \neq K_i \text{ and} \\ & H_K\left(K_* \parallel H^t(a) \parallel H^{Z-t}(b) \parallel ID_i\right) = H_K\left(K_* \parallel H^t(a) \parallel H^{Z-t}(b) \parallel ID_i\right)\} \end{aligned}$$

Because K_i is secret to $\mathcal{A}$, the first probability, Pr_1, is not significantly larger than $1/p$. Recall that p is the order of the elliptic curve subgroup in which we do cryptography. The second probability, Pr_2 is the same as that of finding a collision for the HMAC $H_K(-)$. Both probabilities are negligible. Therefore, it is very unlikely that $\mathcal{A}$ will succeed with a random query message.

The collision resistance of the HMAC also effectively prevents the attacker to correlate the results of multiple random queries to avoid trying points (messages) other than those whose y-coordinates are the same as that of the previously tried points. Therefore the probability of success of the adversary is not significantly better than a brute-force attack.

2.4.3.2 Collusion Attack

We consider the case that $\mathcal{A}$ contains multiple valid users with their assigned devices as well as class keys. These users collude by trying to use their assigned class keys and devices to retrieve a valid temporal key that should not be owned by any of these users. Since it is difficult to combine the HMAC output to infer useful information about the input, we focus on the case that only one device is being queried. We assume this device is associated with class $\mathcal{C}_i$, and it is owned by a member of $\mathcal{A}$.

Assuming the tamper-resistance of the device and the intractability of the discrete logarithm problem, we claim that any collusion attack on the scheme will fail.

Since the encryption information $EncInf_i$ for a device with identity ID_i and the embedded clock cannot be modified because of the tamper-resistance of the device, the device will respond to a single input K_* with a correct decrypting key if and only if $K_* = K_i$. On the one hand, if K_i is one of the attacker's issued class key, then this attempt is a valid regular query, and it will not produce any extra information that the attacker is not supposed to know. On the other hand, if K_i is not owned by $\mathcal{A}$, for an attack to succeed, then it must be derived by $\mathcal{A}$ via collusion, given the infeasibility of guessing, as analyzed in Section 2.4.3.1 above. However, given that all g_j are kept secret, we do not see any way to accomplish this computation without solving the discrete logarithm problem on $E(\mathbb{F}_q)$, even with all $R_{i,j}$ on the public board being available.

Now we consider collusion attacks with more than one input to the device. In this case, $\mathcal{A}$ wants to let the device in class $\mathcal{C}_i$ compute temporal decrypting keys for a class $\mathcal{C}_m$ which is no lower than $\mathcal{C}_i$. Note that such an attack must have $\mathcal{ID}_m$ as one of the three input messages, and carefully choose the other two query messages. The attack inevitably involves the computation (by the device) of the class key K_m. According to Step 2 of the Decrypting Key Derivation process, g_iK_m must be computable by the device in respond to the input parameters. However, we do not know how this computation can be performed, even with the knowledge of K_m, when g_i is unknown.

The analysis in this section implies that it is unlikely that a device is able to effectively compute the temporal keys outside its assigned time period and for classes no lower than itself. Thus it can only function as designated.

2.4.3.3 X. Yi's Attack

As a particular case of the collusion attack just described, X. Yi's attack [21] against H.Y. Chien's scheme [20] cannot be replayed here to break our scheme. We will demonstrate this case to give an impression of how the asymmetry introduced by elliptic curve cryptography helps to strengthen the scheme.

X. Yi's attack can not apply directly to our scheme due to our different construction. The idea of the attack is like this: two users collude to derive certain information

Inf and pass it to a third user, U, so that U can input Inf together with its secret key to the tamper-resistant device to derive the decryption keys of a class no lower than U's. We claim that this analogue of X. Yi's three-party attack does not succeed for our scheme.

Suppose U belongs to class $\mathcal{C}_j$ and U wants to derive decryption keys $K_{i,t}$ of $\mathcal{C}_i$, which is no lower than $\mathcal{C}_j$. Then K_i needs to be computed by the device and passed to the HMAC. An analogue of X. Yi's attack requires the information passed to U be $Inf = [g_j]K_i + ([-1]K_j)$, so that when U inputs Inf, ID_i and K_j, the tamper-resistant device will compute

$$[h_j](Inf + K_j) = [h_j]([g_j]K_i + K_j + [-1]K_j) = K_i.$$

In order to obtain Inf, someone must be able to have knowledge of $g_j K_i$. Given that class $\mathcal{C}_i$ is no lower than $\mathcal{C}_j$, $[g_j]K_i$ is not a summand of any of the published values on the authenticated board, and thus it cannot be produced via collusion, considering the fact that all g_j are secret and the elliptic curve discrete logarithm problem is hard.

Therefore, an obvious generalization of X. Yi's attack cannot be modified to attack our scheme.

2.4.3.4 Remarks on Security Proofs

We want to remark that the analysis above does not provide rigorous security proofs. We do not know yet if choosing suitable input parameters for a device so that it is able to compute a class key belonging to a class no lower than the device's assigned class is equivalent to the elliptic curve discrete logarithm problem. We have not shown rigorously that without knowing the valid class key obtaining a useful decrypting key by querying the device with one input message is equivalent to finding a collision of the HMAC. We do not know if knowing temporal keys is equivalent to knowing the corresponding class key.

We suggest some remaining problems like those above as future research topics. To achieve the security proofs a more formal definition of security will be needed.

2.4.4 Yet Another Good Feature

An important advantage of our scheme is that the vendor can change the class keys of the system at any time without having to re-issue new devices to the users, while only the user's class keys and the public information $R_{i,j}$ need to be updated. In this case, the class keys need to be delivered to users through a secure channel, and the vendor can simply update the database with new values of $R_{i,j}$ on the public board. However, when an individual user wants to change the subscription, a new device needs to be issued. This also needs to be done when a different class is desired.

2.4.5 Space and Time Complexity

Our scheme publishes one value $R_{i,j}$ for each partial order relation $\mathcal{C}_j \prec \mathcal{C}_i$. The total number of public values is at most $\frac{n(n-1)}{2}$, when n is the number of classes in $\mathcal{C}$. On the user's side, the tamper-resistant device stores only H_K, E, $\mathbb{F}_q$, ID_i, h_i and $EncInf_i$.

At any time granule t, the tamper-resistant device needs to perform $(t - t_1) + (t_2 - t) + 2 = t_2 - t_1 + 2 \leq Z$ hash iterations. Note that there are two hash iterations per HMAC operation [26]. In a system of life period 5 years which updates user keys every hour, Z is approximately 43800. We did an experiment using SHA-1 as the hash function on a Gateway MX3215 laptop computer which has a 1.40GHz Intel(R) Celeron(R) M processor, 256 MB of memory and runs Ubuntu 6.10 Edgy Eft. The code is written in C and built with GNU C compiler version 4.1.2. The result showed that 43800 hash iterations took .0800 second of processing time. In practice, $t_2 - t_1$ is usually much smaller than Z and the hash computation is really fast.

The bulk of the computation performed by the tamper-resistant device is the calculation of $K_j = [h_i](R_{i,j}+K_i)$ in Step 2 of the Decrypting Key Derivation phase. A rough estimate [30] shows that a 160-bit prime p (the order of Q on $E(\mathbb{F}_q)$) should give us 80-bit security against the best (generic) elliptic curve discrete logarithm attack in this situation. In this case, the calculation of K_j is comparable to elliptic curve scalar multiplication computation with required precomputation done online. Section 3.7 of [31] gives rough estimates and experimental results for this computational cost, for NIST-recommended curves P-192, B-163 and K-163. The results show that the computation can be performed in several milliseconds on an 800MHz Intel Pentium III using general-purpose registers. A smart card (with a 32-bit processor running at 25 to 32 MHz) can also do this efficiently [32]. Our scheme is in fact slower than H.Y. Chien's scheme, in which only hash computations are widely used. However it is still very efficient from the point of view of application and provides enhanced security.

In Table 2.1 below, we shows a comparison of the three time-bound hierarchical key management schemes.

Table 2.1: A comparison of three time-bound hierarchical key management schemes

Comparison of three schemes			
	Tzeng	Chien	Ours
Implementation requirements	Lucas function	Tamper-resistant device	Tamper-resistant device, ECC
# of public values	$n+6$	$n-1$	$n(n-1)/2$
# of operations to derive temporal secret key of own class	$(t_2-t_1)T_e$, $(t_2-t_1)T_L, T_h$	$(t_2-t_1+1)T_h$	$(t_2-t_1+2)T_h$
# of operations to derive temporal secret key of direct child class	$(t_2-t_1+r)T_e$, $(t_2-t_1)T_L, T_h$	$(t_2-t_1+2)T_h$	$(t_2-t_1+2)T_h, T_E$

# of operations to derive temporal secret key of l-edge-distance child class	$(t_2 - t_1 + r)T_e$, $(t_2 - t_1)T_L, T_h$	$(t_2 - t_1 + 1 + l)T_h$	$(t_2 - t_1 + 2)T_h, T_E$
security against Yi and Ye's attack	insecure	secure	secure
security against X. Yi's attack	N/A	insecure	secure

Suppose $\mathcal{C}_j \preceq \mathcal{C}_i$, $t \in [t_1, t_2]$.

Notation:

n: number of classes $|\mathcal{C}|$

r: number of child classes $\mathcal{C}_i$ on path from $\mathcal{C}_i$ to $\mathcal{C}_j$

T_h: hashing operation

T_e: modular exponentiation

T_L: Lucas function operation

T_E: elliptic curve scalar multiplication

2.5 Future Work and Remarks

Some future directions of this research are:

- Construction of an efficient key management scheme which is provably secure.
- Construction of an efficient key management scheme which does not have to use a tamper-proof device.

- Implementation of the scheme on smart cards and testing.

We want to remark that the choice of the group of an elliptic curve is not intrinsic for this key management scheme: it can be alternatively implemented with small modifications by using any suitable finite group, e.g. the Jacobian of a hyperelliptic curve. In any case, a compact representation of the group element and an efficient group operation must be available. The requirement leads to the subject of the following Chapter 3.

3. ARITHMETIC ON JACOBIANS OF GENUS 2 REAL HYPERELLIPTIC CURVES

3.1 Introduction and Motivation

Since first proposed by N. Koblitz [33] in 1989, the use of the Jacobian of a hyperelliptic curve in public-key cryptography has drawn attention from both academia and industry. With the best known attacks running exponential time, hyperelliptic curves offer a better key-per-bit security compared to conventional schemes like RSA. The attack described in [13] implies that only genus 2 curves provide the same key-per-bit security as elliptic curves. For efficient cryptographic implementation, optimized explicit formulas (e.g. [34]) have been developed for genus 2 imaginary hyperelliptic curves. In [17], a cryptographic key exchange protocol is presented for genus 2 **real** hyperelliptic curves. However, explicit formulas for such curves have not been studied as widely as their imaginary counterparts.

This chapter contains two contributions to the research of explicit formulas for real models of genus 2 hyperelliptic curves:

1. It shows a new result in Theorem 8, which presents an equivalent change of coordinates for a hyperelliptic curve in the real model. This is useful for obtaining a short representation of a curve equation, so that the arithmetic is simplified.

2. The explicit divisor doubling formulas are presented for genus 2 real hyperelliptic curves. These formulas cover all cases of positive characteristic except characteristic 3, for two major representations (i.e. the adapted basis and the reduced basis) of divisors. Explicit formulas for special cases of divisor doubling are also presented. The result shown in Section 3.6 supersedes the doubling for-

mulas found in [35], which deals with characteristic > 3 and divisors represented in the reduced basis.

3.2 Background

For arithmetic purposes, in this chapter, we consider elliptic curves as hyperelliptic curves of genus one. For interests in cryptography, we concentrate on curves over finite fields $\mathbb{F}_q$. We use the definition of hyperelliptic curves by their nonsingular models as follows.

Definition 6 (Hyperelliptic Curves of Genus g, [35]) *A hyperelliptic curve C of genus g defined over $\mathbb{F}_q$ is an absolutely irreducible nonsingular curve defined by an equation*

$$C : y^2 + h(x)y = f(x), \tag{3.1}$$

where $f, h \in \mathbb{F}_q[x]$ are such that $y^2 + h(x)y - f(x)$ is absolutely irreducible; if $b^2 + h(a)b - f(a) = 0$ for $(a, b) \in \bar{\mathbb{F}}_q^{\,2}$, then $2b + h(a) \neq 0$ or $h'(a)b - f'(a) \neq 0$. A hyperelliptic curve is called

1. *an* **imaginary hyperelliptic curve** *if f is monic,* $\deg(f) = 2g + 1$*, and* $\deg(h) \leq g$.
2. *a* **real hyperelliptic curve** *if the following hold: If q is odd, then f is monic,* $h = 0, \deg(f) = 2g+2$. *If q is even, then h is monic,* $\deg(h) = g+1$*, and either (a)* $\deg(f) \leq 2g + 1$ *or (b)* $\deg(f) = 2g + 2$ *and the leading coefficient of f is of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q^*$.*

Let $\mathbb{F}_q(C)$ be the corresponding function field. Let $\mathbb{F}_q[C] = \mathbb{F}_q[x, y]/(y^2 + h(x)y - f(x))$ be the coordinate ring of C. Then $\mathbb{F}_q[C]$ is the integral closure of $\mathbb{F}_q[x]$ in $\mathbb{F}_q(C)$. Let $\mathcal{P}_\infty$ be the place at infinity of $\mathbb{F}_q(x)$. Then an imaginary hyperelliptic curve C corresponds to the case that $\mathcal{P}_\infty$ ramifies in $\mathbb{F}_q(C)$, and a real hyperelliptic curve C

corresponds to the case that $\mathcal{P}_\infty$ splits in $\mathbb{F}_q(C)$ (cf. [36], Chapter 14). We say the curve C has "one point at infinity" or "two points at infinity" accordingly.

Note that according to the above definition, there are models of hyperelliptic curves that are neither imaginary nor real [35] – those that correspond to an inert P_∞. We call such curves *unusual* and exclude such curves from our discussion in this thesis.

Let $Cl(\mathbb{F}_q[C])$ be the ideal class group of the affine algebra $\mathbb{F}_q[C]$. For cryptography, it is worth mentioning the connection between $Cl(\mathbb{F}_q[C])$ and the $\mathbb{F}_q$-rational points of the Jacobian of C, described via the following exact sequences: If C is imaginary, i.e., P_∞ ramifies in $K(C)$, then

$$0 \to Jac_{\mathbb{F}_q}(C) \to Cl(\mathbb{F}_q[C]) \to 0. \tag{3.2}$$

If C is real, i.e., P_∞ splits in $K(C)$, with ∞_1 and ∞_2 above it, then

$$0 \to \langle \infty_1 - \infty_2 \rangle \to Jac_{\mathbb{F}_q}(C) \to Cl(\mathbb{F}_q[C]) \to 0. \tag{3.3}$$

If P_∞ is inert in $K(C)$, then

$$0 \to Jac_{\mathbb{F}_q}(C) \to Cl(\mathbb{F}_q[C]) \to \mathbb{Z}/(2) \to 0.$$

This is a reinterpretation of [36], Propositions 14.6 and 14.7.

3.3 Equivalent Change of Coordinates

Instead of working on curve equations in the most general form, sometimes it is more efficient (and more convenient) to deal with curves given by equations with fewer terms. In this section, we discuss some transformations that can be performed on a curve equation in the general form to obtain an equivalent model of the curve given by a shorter equation. We say two models of a curve are equivalent if they correspond to isomorphic coordinate rings.

3.3.1 Elliptic Curves

It is well-known (see, e.g., [25]) that an elliptic curve given by the (generalized) Weierstraß equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{F}_q, \tag{3.4}$$

under a change of coordinates, is equivalent to a short Weierstraß equation of the form

1. $y^2 = x^3 + Ax + B$, if the characteristic of the field is not 2 or 3;
2. $y^2 = x^3 + a_2'x^2 + a_6'$, if the characteristic of the field is 3 and $j(E) \neq 0$;
3. $y^2 = x^3 + a_4'x + a_6'$, if the characteristic of the field is 3 and $j(E) = 0$;
4. $y^2 + xy = x^3 + a_2'x^2 + a_6'$, if the characteristic of the field is 2 and $a_1 \neq 0$;
5. $y^2 + a_3'y = x^3 + a_4'x + a_6'$, if the characteristic of the field is 2 and $a_1 = 0$.

3.3.2 Imaginary Hyperelliptic Curves

This case occurs when the curve C has an $\mathbb{F}_q$-rational Weierstraß point ∞, i.e., $\dim L(2\infty) > 1$, where $L(2\infty)$ is the Riemann-Roch space of the divisor 2∞. Given a Weierstraß equation of the form

$$y^2 + h(x)y = f(x), \quad \deg(h) \leq g, \deg(f) = 2g + 1, \tag{3.5}$$

the following theorem gives guidelines for obtaining equivalent models.

Theorem 7 (Lockhart, 1994, [37]) *The equation given by (3.5) of a hyperelliptic curve with genus g is unique up to a change of coordinate of the form*

$$x = u^2\hat{x} + r, \quad y = u^{2g+1}\hat{y} + t(\hat{x}),$$

where $u \in \mathbb{F}_q^*, r \in \mathbb{F}_q$ and t is a polynomial over $\mathbb{F}_q$ of degree $\leq g$.

This allows us to justify the isomorphic transformations of imaginary hyperelliptic curves of genus 2 described in [34]:

If $\mathrm{char}(\mathbb{F}_q)$ is odd, then Equation (3.5) can be shortened as $y^2 = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$; if in addition $\mathrm{char}\,\mathbb{F}_q \neq 5$, we further have $f_4 = 0$.

If $\mathrm{char}(\mathbb{F}_q) = 2$, then Equation (3.5) can be transformed to $y^2 + (x^2 + h_1x + h_0)y = x^5 + f_1x + f_0$, or $y^2 + (h_1x + h_0)y = x^5 + f_4x^4 + f_2x^2 + f_1x + f_0$.

3.3.3 Real Hyperelliptic Curves

This case occurs when the infinite place of the subfield $\mathbb{F}_q(x)$ of $\mathbb{F}_q(C)$ splits and the curve is given by the model

$$y^2 + h(x)y = f(x), \quad \deg(h) \leq g + 1, \deg(f) \leq 2g + 2. \tag{3.6}$$

We adapt the statement and proof of Theorem 7 to the case of real hyperelliptic curves, and present them as follows.

Theorem 8 *The equation given by (3.6) of a hyperelliptic curve is unique up to a change of coordinates of the form*

$$x = u\hat{x} + r, \quad y = \pm u^{g+1}\hat{y} + t(\hat{x}),$$

where $u \in \mathbb{F}_q^*, r \in \mathbb{F}_q$ *and* t *is a polynomial over* $\mathbb{F}_q$ *with* $t = 0$ *if* q *is odd, and* $\deg(t) \leq g + 1$ *if* q *is even.*

Proof Let us first briefly review how Equation (3.6) is obtained. Let P_∞ be the place at infinity of $\mathbb{F}_q(x)$, which splits in $\mathbb{F}_q(C)$ with $\{\infty_1, \infty_2\}$ lying above. Then by construction, we have that the Riemann–Roch space $L(P_\infty)$ has basis $\{1, x\}$. For $1 \leq j \leq g$ we have $\dim L(jP_\infty) \geq 2j - g + 1$, by the Riemann-Roch theorem, and the elements $\{1, x, x^2, \ldots, x^j\}$ are linearly independent over $\mathbb{F}_q$ in $L(jP_\infty)$. We also have $\dim L\left(gP_\infty\right) = \deg\left(gP_\infty\right) - g + 1 = g + 1$, $\dim L\left((g+1)P_\infty\right) = \deg\left((g+1)P_\infty\right) - g + 1 = g + 3$, for $\deg\left(gP_\infty\right) = 2g > 2(g-1)$ and $\deg\left((g+1)P_\infty\right) = 2g + 2 > 2(g-1)$. The $g+1$ functions $1, x, x^2, \ldots, x^g$ form a basis for $L(gP_\infty)$.

The function $x^{g+1} \in L((g+1)P_\infty) \backslash L(gP_\infty)$. To form a basis for $L((g+1)P_\infty)$, there must be another function, y, which is linearly independent of the powers of x, in $L((g+1)P_\infty) \backslash L(gP_\infty)$. Now we look at $L(2(g+1)P_\infty)$, which is $(3g+5)$–dimensional and contains the $3g+6$ functions

$$1, x, x^2, \ldots, x^{g+1}, y, x^{g+2}, xy, \ldots, x^{2g+2}, x^{g+1}y, y^2. \tag{3.7}$$

Therefore there is a nontrivial $\mathbb{F}_q$–linear relationship among them. Since the functions of powers of x in (3.7) are linearly independent over $\mathbb{F}_q$ and $y \notin K[x]$, the coefficient of y^2 must not be 0. Multiplying the linear relation with the multiplicative inverse of the coefficient of y^2 we obtain the model of the curve in the form of (3.6). Furthermore, W.L.O.G[1], we may assume the conditions on coefficients and degrees as in Definition 6 are satisfied.

Now suppose $\hat{x}$ and $\hat{y}$ are another such pair of functions as x and y above. Then $\hat{x} \in L(P_\infty)$, and we must have $x = a\hat{x} + r$ for some $a \in \mathbb{F}_q^*$, $r \in \mathbb{F}_q$. Similarly, $\hat{y} \in L((g+1)P_\infty)$, and thus we have $y = b\hat{y} + t(\hat{x})$ for $b \in \mathbb{F}_q^*$ and $t(\hat{x}) \in \mathbb{F}_q[\hat{x}]$, $\deg(t) \leq g+1$.

If q is odd, then the monicity of the coefficients for $\hat{x}^{2g+2}$ and $\hat{y}^2$ and degeneracy of the term $\hat{x}\hat{y}$ implies that $b^2 = a^{2g+2}$ and $t(x) = 0$. Let $u = a$. Then $u \in \mathbb{F}_q^*$ and $b = \pm u^{g+1}$.

If q is even, then the model in x and y is in the form of (3.6) with h monic of degree $g+1$, and either (a) $\deg(f) \leq 2g+1$, or (b) $\deg(f) = 2g+2$ and f has a leading coefficient $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q^*$. Let t_1 be the coefficient of $\hat{x}^{g+1}$ in $t(\hat{x})$. We look at the coefficients of the terms in $\hat{x}$ and $\hat{y}$ with pole order $2g+2$ at ∞_1 (or ∞_2). In case (a), we have $b^2\hat{y}^2 + a^{g+1}b\hat{x}^{g+1}\hat{y} = (t_1^2 + a^{g+1}t_1)\,\hat{x}^{2g+2}$, i.e., $\hat{y}^2 + (a^{g+1}/b)\hat{x}^{g+1}\hat{y} = (1/b^2)\,(t_1^2 + a^{g+1}t_1)\,\hat{x}^{2g+2}$. Therefore, we must have $a^{g+1}/b = 1$, i.e., $b = a^{g+1}$. Let $u = a$. Then $b = u^{g+1}$. And either the coefficient of $\hat{x}^{2g+2}$ is 0, if $t_1 = 0$, or it is equal to $z^2 + z$, where $z = t_1/u^{g+1} \in \mathbb{F}_q^*$ with $t_1 \neq 0$. Similarly, in case (b), we let $u = a$. Then we have $b = u^{g+1}$ and $\hat{x}^{2g+2}$ has coefficient of the form $z^2 + z$, where $z = \beta + t_1/b$. This completes the proof. ■

---

[1] If q is odd, make the change of variable $y \leftarrow y - h(x)/2$; if q is even, cf. [38], Theorem 7.

A genus 2 real hyperelliptic curve C over a finite field $\mathbb{F}_q$ can be given by the equation

$$y^2 + h(x)y = f(x),$$

where $h(x) = (h_3x^3+h_2x^2+h_1x+h_0)$, $f(x) = f_6x^6+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$, and $h_i, f_j \in \mathbb{F}_q$.

If $\mathrm{char}(\mathbb{F}_q)$ is odd, then $h = 0$ and $f_6 = 1$. In particular, if $\mathrm{char}(\mathbb{F}_q) > 3$, $f(x)$ can be written as $f(x) = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$ with a linear change of variable $x \leftarrow x + f_5/6$. This shorter equation is equivalent to the original one, in the sense that they give the same coordinate ring.

If $\mathrm{char}(\mathbb{F}_q) = 2$, then $h(x)$ is monic, $\deg(h) = 3$, and either $\deg(f) \leq 5$, or $\deg(f) = 6$ and f_6 is of form $\beta^2 + \beta \neq 0$ for some $e \in \mathbb{F}_q^*$.

Now suppose C is written in the form

$$y^2 + (x^3 + h_2x^2 + h_1x + h_0)y = f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0,$$

where $f_6 = e^2 + e \in \mathbb{F}_q$, which can be zero or nonzero. The change of variable $x \leftarrow x + h_2$ makes the h_2 term vanish.

$$C : y^2 + (x^3 + h_1x + h_0)y = f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0.$$

Then $y \leftarrow y + f_5x^2$ eliminates the f_5 term.

$$C : y^2 + (x^3 + h_1x + h_0)y = f_6x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0.$$

Then $y \leftarrow y + f_4x$ eliminates the f_4 term

$$C : y^2 + (x^3 + h_1x + h_0)y = f_6x^6 + f_3x^3 + f_2x^2 + f_1x + f_0.$$

Then $y \leftarrow y + f_3$ eliminates the f_3 term

$$C : y^2 + (x^3 + h_1x + h_0)y = f_6x^6 + f_2x^2 + f_1x + f_0.$$

This is the shortest Weierstraß equation we can use.

3.4 Explicit Formulas for Elliptic and Genus 2 Imaginary Hyperelliptic Curves

3.4.1 Mumford Representation and Cantor's algorithm

In order to use the Jacobian of a hyperelliptic curve in cryptography, we must have a compact encoding of its elements as well as efficient arithmetic operations on the elements. For imaginary hyperelliptic curves, the Mumford representation of the degree 0 divisor class and Cantor's algorithm provide the facilities to do so.

Theorem 9 (Mumford representation, cf. [14, 39]) *Let C be a genus g hyperelliptic curve given by $C : y^2 + h(x)y = f(x)$, where $h, f \in \mathbb{F}_q[x], \deg(f) = 2g + 1, \deg(h) \leq g$. Each nontrivial divisor class over $\mathbb{F}_q$ can be represented via a unique pair of polynomials $u(x)$ and $v(x)$, $u, v \in \mathbb{F}_q[x]$, where*

1. *u is monic,*
2. *$\deg(v) < \deg(u) \leq g$,*
3. *$u | v^2 - vh - f$.*

Let $D = \sum_{i=1}^{r} P_i - r\infty$, where $P_i \neq \infty$, $P_i \neq -P_j$ for $i \neq j$ and $r \leq g$. Put $P_i = (x_i, y_i)$. Then the divisor class of D is represented by

$$u(x) = \sum_{i=1}^{r}(x - x_i)$$

and if P_i occurs n_i times then

$$\left(\frac{d}{dx}\right) \left[v(x)^2 - v(x)h(x) - f(x)\right] |_{x=x_i} = 0, \text{ for } 0 \leq j \leq n_i - 1.$$

Such a pair $[u, v]$ is called a Mumford representation of divisor class of the curve C (or ideal class of the corresponding affine coordinate ring $\mathbb{F}_q[C]$).

The Mumford representation is usually defined for elements of the Jacobian of an imaginary hyperelliptic curve, as described above. For real hyperelliptic curves, an analogous definition is also achievable for the *infrastructure* within the Jacobian of the curve. We will expose more on it in Section 3.5.

As an analogue of the composition of binary quadratic forms originated by Gauss, the Cantor's algorithm [33,40] realizes the hyperelliptic curve Jacobian group law by working on the Mumford representation of the elements.

Algorithm 1 Cantor's algorithm

Input: Two divisor classes $\bar{D}_1 = [u_1, v_1]$ and $\bar{D}_2 = [u_2, v_2]$ on the curve $C : y^2 + h(x)y = f(x)$.

Output: The unique reduced divisor $D = [U, V]$ such that $\bar{D} = \bar{D}_1 \oplus \bar{D}_2$.

```
$d_1 \leftarrow \gcd(u_1, u_2)$; $d_1 = e_1 u_1 + e_2 u_2$
$d \leftarrow \gcd(d_1, h - v_1 - v_2)$; $d = c_1 d_1 + c_2(h - v_1 - v_2)$
$s_1 \leftarrow c_1 e_1, s_2 \leftarrow c_1 e_2$ and $s_3 \leftarrow c_2$
$U \leftarrow u_1 u_2 / d^2$ and $V \leftarrow (s_3(v_1 v_2 + f) - s_1 u_1 v_2 - s_2 u_2 v_1)/d \pmod{U}$
repeat
    $U' \leftarrow (f + Vh - V^2)/U$ and $V' \leftarrow h - V \pmod{U'}$
    $U \leftarrow U'$ and $V \leftarrow V'$
until $\deg(U) \leq g$
make $U$ monic and return $[U, V]$
```

Again, Cantor's algorithm is designed to work for hyperelliptic curve in the imaginary model, but it can be modified for the infrastructure of real hyperelliptic curves. We will have more discussion in Section 3.5.

3.4.2 Elliptic Curves And Genus 2 Hyperelliptic Curves in the Imaginary Model

An elliptic curve E with a point ∞ at infinity is isomorphic to its degree 0 divisor class group $Pic^0(C)$. Applying Cantor's algorithm to E, the well–known group law can be explicitly described as follows.

Theorem 10 (Group Law Algorithm 2.3, [41]) *Let E be an elliptic curve given by a Weierstraß equation*

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

(a) Let $P_0 = (x_0, y_0) \in E$. Then

$$-P_0 = (x_0, -y_0 - a_1x_0 - a_3).$$

Now let

$$P_1 + P_2 = P_3 \quad \textit{with} \quad P_i = (x_i, y_i) \in E.$$

(b) If $x_1 = x_2$ and $y_1 + y_2 + a_1x_2 + a_3 = 0$, then

$$P_1 + P_2 = \infty.$$

Otherwise let

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \mu = \frac{y_1x_2 - y_2x_1}{x_2 - x_1} \quad \textit{if } x_1 \neq x_2;$$

$$\lambda = \frac{2x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3},$$

$$\mu = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3} \quad \textit{if } x_1 = x_2.$$

(c) $P_3 = P_1 + P_2$ is given by

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2,$$

$$y_3 = -(\lambda + a_1)x_3 - \mu - a_3.$$

(d) As special cases of (c), we have for $P_1 \neq \pm P_2$,

$$x(P_1 + P_2) = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 + a_1\left(\frac{y_2 - y_1}{x_2 - x_1}\right) - a_2 - x_1 - x_2;$$

and the doubling formula for $P = (x, y) \in E$,

$$x([2]P) = \frac{x^4 - b_4x^2 - 2b_6x - b_8}{4x^3 + b_2x^2 + 2b_4x + b_6},$$

where

$$b_2 = a_1^2 + 4a_2, \quad b_4 = 2a_4 + a_1a_3, \quad b_6 = a_3^2 + 4a_6 \quad \textit{and}$$

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

The Riemann Hypothesis for abelian varieties over finite fields (cf. [42]) implies that the cardinality of the rational points of the Jacobian variety of a genus g curve over a finite field $\mathbb{F}_q$ is approximately q^g. This is q^2 for the case of genus 2. It is the 1-1 correspondence, as shown in (3.2), between the categories of divisor class groups of imaginary genus 2 hyperelliptic and ideal class groups of the coordinate ring $\mathbb{F}_q[C]$ of the curve that makes the arithmetic on divisor class groups well understood. Thus it becomes clear how to do discrete logarithm based cryptography on Jacobians of imaginary genus 2 curves. Given that the best attacks to the discrete logarithm problem in a genus 2 Jacobian are the generic ones that have complexity $O(\sqrt{N})$, where N is the group order (cardinality), the use of Jacobians of genus 2 curves for discrete logarithm based cryptography versus elliptic curves (genus 1 case), though more complicated, has the advantage that to achieve the same level of security, it requires a smaller size of the underlying finite field in which we do the actual arithmetic. Genus 2 curves for cryptography are more suitable for smaller processor architectures, and **may** potentially offer us extra efficiency, compared to their elliptic curve counterpart [43]. The rigorous argument cannot be easily made; it is complicated by many factors, including design and implementation. To make such an estimate feasible, a first and very important step is to understand better the divisor arithmetic for genus 2 curves.

Following T. Lange's pioneering work [34], extensive efforts have been put into the research on explicit formulas, i.e., formulas written in terms of actually field arithmetic, for imaginary genus 2 hyperelliptic curves of the following form:

$$C : y^2 + h(x)y = f(x), \tag{3.8}$$

where $f(x)$ is a degree 5 polynomial defined over the finite field $\mathbb{F}_q$, and $\deg(h) \leq 2$.

Such explicit formulas are computationally more efficient than the generic algorithm (the Cantor's algorithm) given in polynomial operations, for performing the group law in the Jacobian of genus 2 curves. The following Tables 3.1 and 3.2 show a performance comparison of the generic algorithm and the explicit formulas. The code is written in C++ and built over the `libg2hec` genus 2 crypto library [44] using the gcc compiler version 4.2.3 with flag `-O2`. The `libg2hec` library uses field arithmetic provided by V. Shoup's NTL library [45], and implements the Cantor's algorithm as in Algorithm 1 and the explicit formulas for imaginary genus 2 curves in affine coordinates. The experiment was done on a Lenovo ThinkPad T61 laptop computer which has an Intel(R) Core 2 Duo processor (T7300 2.0GHz), 2 GB RAM, and runs Linux kernel version 2.6.24-19.

Table 3.1: Performance Comparison: average time (in ms) for one addition/doubling over prime fields of size ℓ (in bits)

Size ℓ (in bits)	Cant. add	Expli. add	Cant. doub.	Expli. doub.
80	0.07608	0.05528	0.08232	0.06728
160	0.09956	0.07396	0.10912	0.08948
256	0.13248	0.103	0.14564	0.1246
324	0.16784	0.13092	0.18616	0.15976
512	0.23188	0.17708	0.26068	0.21864
1024	0.52108	0.37364	0.57692	0.45828

Table 3.2: Ratio of Operational Times

Size ℓ (in bits)	80	160	256	324	512	1024
Cant./Expli. add	1.37627	1.34613	1.28621	1.282	1.30946	1.3946
Cant./Expli. doub.	1.22354	1.21949	1.16886	1.16525	1.19228	1.25888

3.5 Genus 2 Hyperelliptic Curves in Real Model

We shall consider another type of genus 2 curve, i.e., *real genus 2 hyperelliptic curves*. Recall that they have the form

$$C : y^2 + h(x)y = f(x), \tag{3.9}$$

where $f, h \in \mathbb{F}_q[x]$ are such that $y^2 + h(x)y - f(x)$ is absolutely irreducible, i.e. irreducible over $\mathbb{F}_q$, and if $b^2 + h(a)b = f(a)$ for $(a, b) \in \mathbb{F}_q \times \mathbb{F}_q$, then $2b + h(a) \neq 0$ or $h'(a)b - f'(a) \neq 0$. If q is odd, then f is monic, $h = 0, \deg(f) = 6$. If q is even, then h is monic, $\deg(h) = 3$, and either (a) $\deg(f) \leq 5$ or (b) $\deg(f) = 6$ and the leading coefficient of f is of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q^*$.

A genus 2 curve of this form has two "points at infinity." Unlike the case of imaginary genus 2 hyperelliptic curves, the divisor class group of a real genus 2 hyperelliptic curve does not correspond nicely to the ideal class group of the coordinate ring $\mathbb{F}_q[C]$ of the curve (cf. (3.3) or [36], Proposition 14.7). Moreover, as in the number field case, each ideal class of the function field K is not uniquely represented by reduced ideals. This creates obstacles for using the ideal class group of a real hyperelliptic curve for computation, hence limits the use of real hyperelliptic curves in discrete logarithm based cryptography. However, the *infrastructure* of the divisor class group leads us to look at the problem from a different perspective. The following theorem sets up a connection between the divisor class group and the set of reduced ideals of $\mathbb{F}_q(C)$.

Theorem 11 (Paulus-Rück, 1999, [46]) *There is a canonical bijection between the divisor class group* $Jac_{\mathbb{F}_q}(C)$ *and the set of pairs* $\{(\mathfrak{a}, n)\}$*, where* $\mathfrak{a}$ *is a reduced ideal of* $\mathbb{F}_q[C]$ *and* n *is a non-negative integer with* $0 \leq \deg(\mathfrak{a}) + n \leq g$*.*

For arithmetic purposes, we restrict our attention to the special subset $\mathcal{R} := \{(\mathfrak{a}, 0) : \mathfrak{a} \text{ reduced and principal}\}$ of the Jacobian (up to isomorphism). Define the *regulator* R of $\mathbb{F}_q(C)$ in $\mathbb{F}_q[C]$ to be the order of $\infty_1 - \infty_2$ in $Jac_{\mathbb{F}_q}(C)$. Recall from (3.3) the group isomorphism

$$Jac_{\mathbb{F}_q}(C)/\langle \infty_1 - \infty_2 \rangle \simeq Cl(\mathbb{F}_q[C]).$$

Now it is clear that $\#\mathcal{R} \leq R$.

$\mathcal{R}$ can be made into a totally ordered set with an order introduced by a *distance function* $\delta(\cdot)$ as follows: Fix the trivial ideal $\mathfrak{a}_1 = (1) = \mathbb{F}_q[C] \in \mathcal{R}$. For any ideal $\mathfrak{b} \in \mathcal{R}$, by definition, there exists $\alpha \in \mathbb{F}_q(C)^*$ such that $\mathfrak{b} = (\alpha)\mathfrak{a}_1$. Let $\delta(b) = -\nu_1(\alpha) \pmod{R}$, where ν_1 is the normalized valuation of $\mathbb{F}_q(C)$ at ∞_1.[2] It follows that elements in R are uniquely determined by their distances, and that $\mathcal{R}$ is a totally ordered set by distance [47]. More precisely, we can write $\mathcal{R} = \{\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_m\}$, where $m \leq R$ and $0 = \delta(\mathfrak{a}_1) < \delta(\mathfrak{a}_2) < \ldots < \delta(\mathfrak{a}_m) < R$. Given the one-to-one correspondence discussed in Theorem 11, this can also be written as

$$\mathcal{R} = \{\bar{D}_1, \bar{D}_2, \ldots, \bar{D}_m\}, \quad \text{where } m \leq R, \text{and } 0 = \delta(\bar{D}_1) < \ldots < \delta(\bar{D}_m) < R,$$

in divisor class notation. The set $\mathcal{R}$ with the distance function $\delta(\cdot)$ is called the *infrastructure* of the principal ideal class.

An algorithm for performing arithmetic operations in $\mathcal{R}$ has been presented in [17, 46, 48]. It consists of three steps:

(a) composition of reduced ideals,

(b) reduction of the primitive part of the product, and

(c) baby steps, i.e. adjusting the output of the reduction so that the degree condition of the theorem is satisfied.

Step (a) and (b) together are called a *giant step*. A giant step, often written as $\mathfrak{a}_i \cdot \mathfrak{a}_j$ (or equivalently, $\bar{D}_i + \bar{D}_j$), is the analogue of the group operation in the imaginary case. A baby step is unique to real hyperelliptic curves. As described in [35], it is the operation $\mathfrak{a}_i \rightarrow \mathfrak{a}_{i+1}$ (or equivalently, $\bar{D}_i \rightarrow \bar{D}_{i+1}$).

---

[2] The normalized valuation ν_1 takes value -1 at x.

Note that $\mathcal{R}$ is not a group with respect to the giant step operation, because associativity does not necessarily hold [49]. However, it is true that for $D, D' \in \mathcal{R}$ we have

$$\delta(D + D') = \delta(D) + \delta(D') - d, \text{where } 0 \leq d \leq 4,$$

and d can be efficiently computed (see, for example, Section 2 of [49]). This allows us to do cryptography in $\mathcal{R}$. Several key-exchange protocols based on arithmetic in $\mathcal{R}$ have been presented in [49]. The need for efficient implementation of these protocols promotes investigations on explicit arithmetic in divisor class groups of real genus 2 hyperelliptic curves.

Our research focuses on the *explicit formulas*: we use the (affine) *Mumford representation* (defined in Section 3.6 below) of the divisor classes of real hyperelliptic curves, apply the *composition* and *reduction* algorithms for operations in $\mathcal{R}$, work on finite field arithmetic directly to derive the baby step and giant step (divisor addition and doubling) formulas, and optimize the results. We only present the result on divisor doubling formulas for the giant step in detail in this thesis. Further information about the baby step and addition formulas can be found in [35, 50, 51]

3.6 Explicit Formulas for the Real Model

We now restrict our attention to the arithmetic of the subset of $Jac_{\mathbb{F}_q}(C)$ that corresponds to $\mathcal{R}$. Note that any element in $\mathcal{R}$ corresponds uniquely to a reduced principal ideal $\mathfrak{a}$ of $\mathbb{F}[C]$, which can be represented by a pair $[u, v]$ of polynomials. Therefore, we only look at and perform operations on elements of $Jac_{\mathbb{F}_q}(C)$ which are given by a pair $\bar{D} = [u, v]$ of polynomials over $\mathbb{F}_q$, such that

1. u is monic,
2. $\deg(u) \leq g$,
3. $u | v^2 - vh - f$,
4. one of the following conditions is satisfied:

(a) for the *adapted (standard) basis*: $\deg(v) < \deg(u)$, or

(b) for the *reduced basis*: $-\nu_1(v-h-y) < -\nu_1(u) < -\nu_1(v+y)$.

If only 1, 3, and 4 are satisfied, $\bar{D}$ is called *semi-reduced.* If all four conditions are satisfied, then $\bar{D}$ is called *reduced.* For details on the above representation, see [35,47].

This is the Mumford representation of divisor classes in the infrastructure $\mathcal{R}$ (in adapted/reduced basis). As with Cantor's algorithm in the case of imaginary hyperelliptic curves, operation can be applied to $\mathcal{R}$ to compose or reduce divisors, and thus implement the baby step and giant step operations. The algorithms are introduced in [35]. We describe them in Algorithms 2 and 3, as follows:

```
Algorithm 2 Composition
Input: Two divisor classes $\bar{D}_1 = [u_1, v_1]$ and $\bar{D}_2 = [u_2, v_2]$ on the curve $C : y^2 + h(x)y = f(x)$.
Output: A semi-reduced divisor $D = [U, V]$ such that $\bar{D} = \bar{D}_1 \oplus \bar{D}_2$.
$d \leftarrow \gcd(u_1, u_2, v_1 + v_2 + h) = x_1u_1 + x_2u_2 + x_3(v_1 + v_2 + h)$
$U \leftarrow u_1u_2/d^2$ and $V \leftarrow (x_3(v_1v_2 + f) + x_1u_1v_2 + x_2u_2v_1)/d \pmod{U}$
```

Let $H(x) = \lfloor y \rfloor$ be the principal part of a root y of $y^2 + h(x)y - f(x) = 0$, i.e., if $y = \sum_{i=-\infty}^{m} y_i x^i \in \mathbb{F}_q\langle x^{-1}\rangle$, then $H(x) = \sum_{i=0}^{m} y_i x^i$.

```
Algorithm 3 Reduction
Input: A divisor class $\bar{D} = [u, v]$ the curve $C : y^2 + h(x)y = f(x)$.
Output: A reduced divisor $D' = [U, V]$ such that $\bar{D}' = \bar{D}$.
$a \leftarrow (v + H(x))$ div $u$.
repeat
    $V \leftarrow v - au, U \leftarrow (f + hV - V^2)/u$.
until $\deg(U) \leq g$
Make $U$ monic and adjust $V$ to a adapted/reduced basis if necessary.
```

Let $[u, v]$ be a Mumford representation of a divisor class in $\mathcal{R}$. We discuss explicit formulas for divisor class addition (ideal multiplication), divisor class doubling (ideal squaring), and a baby step (ideal reduction). We present doubling formulas in detail. The formulas are presented for both the adapted and reduced bases.

While the formulas are given for the hyperelliptic curve in the most general form

$$C : y^2 + h(x)y = f(x), \tag{3.1}$$

where $h(x) = h_3x^3 + h_2x^2 + h_1x + h_0$ and $f(x) = f_6x^6 + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$, we make the following assumptions when counting the number of finite field operations.

- If the characteristic of the field is a prime $p > 3$, we can transform the general equation defining the curve to one of the form

  $$C : y^2 = f(x)$$

  that is equivalent to the original curve, where $f(x) = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$. i.e., we can assume that $h(x) = 0$, the leading coefficient of $f(x)$ is 1, and the x^5 term of $f(x)$ is 0.

- If the characteristic of the field is 2, we consider the hyperelliptic curve given by an equation of the form

  $$C : y^2 + h(x)y = f(x),$$

  where $f(x) = f_6x^6 + f_2x^2 + f_1x + f_0$, and $h(x) = x^3 + h_1x + h_0$. In this case, f_6 is of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q^*$. We ignore the count for multiplications by f_6 or β in explicit formulas.

With the assumptions above, we write $y = \sum_{i=-\infty}^{m} y_ix^i \in \mathbb{F}_q\langle x^{-1}\rangle$, substitute it into the curve equation, determine the values y_i by comparing coefficients of powers of x, and pre-compute $H(x) = y_3x^3 + y_2x^2 + y_1x + y_0$ as follows. If the finite field has odd characteristic and

$$f(x) = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0 \qquad h(x) = 0,$$

we have $y_3 = 1$, $y_2 = 0$, $y_1 = f_4/2$ and $y_0 = f_3/2$. Therefore a reduced basis $[u, v]$ in this case has v of the form $v(x) = x^3 + v_1 x + v_0$. If the finite field has an even characteristic and

$$f(x) = (\beta^2 + \beta)x^6 + f_2 x^2 + f_1 x + f_0, \quad h(x) = x^3 + h_1 x + 1,$$

we have $y_3 = \beta'$, $y_2 = 0$, $y_1 = \beta' h_1$ and $y_0 = \beta' h_0$, where $\beta' = \beta$ or $\beta + 1$. Note that in this case $H(x) = \beta' h(x)$. In this case, a reduced basis $[u, v]$ has v of the form $(1 + \beta')x^3 + v_1 x + v_0$.

3.6.1 Baby Step and Addition Formulas

We do not present in this thesis the baby step and addition formulas, which will be available in [50]. Partial results of the current research is summarized in Table 3.11.

3.6.2 Doubling Formulas

3.6.2.1 Algorithm for Divisor Doubling

Let $[u, v]$ be a degree 2 reduced divisor written in the Mumford representation, with both points of the divisor not equal to their opposites. To perform divisor doubling, compute the following expressions. Similar formulas for simplification of the Cantor's algorithm for genus 2 imaginary hyperelliptic curves were first suggested by R. Harley in [52] and extended by T. Lange [39] et. al. We will show that these formulas give the desired result.

$$\begin{array}{ll} \widetilde{v} \equiv 2v - h \pmod{u} & r = \text{resultant}(u, \widetilde{v}) \\ inv \equiv r(\widetilde{v})^{-1} \pmod{u} & k = (f + hv - v^2)/u \\ s' \equiv k \cdot inv \pmod{u} & s = \dfrac{1}{r} \cdot s' \\ \widetilde{u} = s^2 + ((2v - h)s - k)/u & u' = \widetilde{u} \text{ made monic} \\ \quad \text{Adapted basis:} & v' = h - su - v \pmod{u'} \\ \quad \text{Reduced basis:} & v' = H(x) + h(x) - [(H(x) + s \cdot u + v) \pmod{u'}] \end{array}$$

Let $[u, v] = [x^2 + u_1x + u_0, x^3 + v_1x + v_0]$ be a degree two reduced basis Mumford representation with both points of the divisor not equal to their opposites. Then Cantor's Algorithm for doubling the divisor (u, v) must result in (U_1, V_1) such that

$$\begin{aligned} &U_0 = u^2 \\ &V_0 \equiv v \pmod{u} \\ &(V_0 = v + su \text{ for some } s) \\ &V_1 = h - V_0 + \left\lfloor \tfrac{V_0 + H(y)}{U_0} \right\rfloor U_0 \\ &U_1 = (f + h \cdot V_1 - V_1^2)/U_0 \end{aligned}$$

Here, s is chosen such that U_0 divides $V_0^2 - h \cdot V_0 - f$. Again, $\left\lfloor \frac{V_0 + H(x)}{U_0} \right\rfloor$ is zero since U_0 has degree 4 and $V_0 + H(y)$ has degree 3. Hence, $V_1 = h - V_0 = h - su - v$ and

$$\begin{aligned} U_1 &= \left(f + h \cdot (h - su - v) - (h - su - v)^2\right)/u^2 \\ &= \left(f + hv - v^2 - us(2v - h) - s^2u^2\right)/u^2 \\ &= \left((f + hv - v^2)/u - (2v - h)s\right)/u - s^2 \\ &= (k - (2v - h)s)/u - s^2 \end{aligned}$$

where the division in $k = (f + hv - v^2)/u$ is exact. To make the division of $k - (2v - h)s$ by u exact, we choose $s \equiv k \cdot (2v - h)^{-1} \pmod{u}$, and obtain

$$k - (2v - h)s \equiv k - (2v - h) \cdot k \cdot (2v - h)^{-1} \equiv 0 \pmod{u} \ .$$

Finally, we reduce $[U_1, V_1]$ into either adapted or reduced basis. U_1 is made monic to yield

$$u' = s^2 + (2vs - k)/u \text{ made monic .}$$

For the adapted basis, $v' = V_1 \bmod u'$. For the reduced basis, $v' = H(x) + h(x) - [H(x) + h(x) - V_1 \bmod u'] = H(x) + h(x) - [(H(x) + su + v) \bmod u']$.

3.6.2.2 General Explicit Formulas for Divisor Doubling

Instead of following the doubling formulas described in Section 3.6.2.1 with polynomial operations, we write the result of each step in terms of finite field operations, optimize manually using techniques like Karatsuba multiplication, subexpression elimination, etc., and derive the **explicit formulas** for divisor doubling in this chapter.

We only count inversions, multiplications and squarings of finite field elements, which consist of the bulk of the computation when compared with addition and subtractions. In the tables below, we denote finite field "inversion", "multiplication," and "squaring" by I, M, and S, respectively.

The resulting explicit formulas are presented in Tables 3.3, 3.4 and 3.5.

Throughout the chapter, the total number of operations in parentheses are for the case of even characteristic.

Table 3.3: Explicit Formulas for Doubling Divisor Classes

Doubling, General Case, $\deg u = 2$	
Input	$[u, v], u = x^2 + u_1x + u_0, v = v_3x^3 + v_2x^2 + v_1x + v_0$
Output	$[u', v'] = 2[u, v] := [u, v] + [u, v]$

Step	Expression	adapted	reduced
1	$\underline{\widetilde{v} = \widetilde{v}_1x + \widetilde{v}_0}$ $w_1 = u_1^2$, $t_i = 2v_i - h_i$, $i = 0, 1, 2, 3$	S (S, M)	S,M
Continued on next page			

Table 3.3 – continued from previous page

Step	Expression	adapted	reduced
	$\widetilde{v}_1 = t_3(w_1 - u_0) - t_2u_1 + t_1$, $\widetilde{v}_0 = u_0 \cdot (t_3u_1 - t_2) + t_0$		
2	$\underline{r = res(\widetilde{v}, u),\ inv = inv_1 x + inv_0}$ $w_2 = u_0 \cdot \widetilde{v}_1$, $w_3 = u_1 \cdot \widetilde{v}_1$, $inv_1 = -\widetilde{v}_1$, $inv_0 = \widetilde{v}_0 - w_3$, $r = \widetilde{v}_0 \cdot inv_0 + \widetilde{v}_1 \cdot w_2$	4M	4M
3	$\underline{k' \equiv (f + hv - v^2)/u \pmod{u} = k'_1x + k'_0}$: $k'_4 = f_6 + v_3(h_3 - v_3)$, $k'_3 = f_5 - t_2v_3 + h_3v_2 - 2k'_4u_1$, $k'_2 = f_4 - t_1v_3 + v_2(h_2 - v_2) + h_3v_1 - 2k'_3u_1 - k'_4(w_1 + 2u_0)$, $k'_1 = f_3 - t_0v_3 - t_1v_2 + h_2v_1 + h_3v_0 - 2u_1(k'_2 + k'_4u_0) - k'_3(w_1 + 2u_0)$, $k'_0 = f_2 - t_0v_2 + v_1(h_1 - v_1) + h_2v_0 - (k'_1 + 2k'_3u_0)u_1 - k'_2(w_1 + 2u_0) - k'_4u_0^2$	2S, 3M (2S, 2M)	S, 3M (3M)
4	$\underline{s' = s'_1x + s'_0}$ $s'_1 = \widetilde{v}_0 \cdot k'_1 - \widetilde{v}_1 \cdot k'_0$, $s'_0 = w_2 \cdot k'_1 + inv_0 \cdot k'_0$	4M	4M
$\surd$	Set $r_2 = r^2$. Check conditions to see if $\widehat{w}_0$ will be 0 in Step 5 below; if it will, see Doubling Special Cases: Table 3.4 or Table 3.5.	S (S, M)	S
5	$\underline{\text{Inversion, } r^{-1},\ s_0,\ s_1,\ \widetilde{u}_2^{-1}}$ $\widehat{w}_0 = s'_1\cdot(t_3r+s'_1)-k'_4r_2(= r^2\widetilde{u}_2)$, $\widehat{w}_1 = (r\cdot\widehat{w}_0)^{-1}$ $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1(= \frac{1}{r})$, $\widehat{w}_3 = r \cdot r_2 \cdot \widehat{w}_1(= \frac{1}{\widetilde{u}_2})$ $s_1 = \widehat{w}_2 \cdot s'_1$, $s_0 = \widehat{w}_2 \cdot s'_0$	I, S, 6M (I, 6M)	I, 7M
6	$\underline{u' = x^2 + u'_1x + u'_0}$	S, 4M	5M

Continued on next page

Table 3.3 – continued from previous page

Step	Expression	adapted	reduced
	$u'_1 = \widehat{w}_3 \cdot (s_1 \cdot (-u_1t_3 + t_2 + 2s_0) + s_0t_3 - k'_3)$ $u'_0 = \widehat{w}_3 \cdot (s_1 \cdot \widetilde{v}_1 + s_0 \cdot (-u_1t_3 + t_2 + s_0) - k'_2)$	(5M)	
7	$\underline{v' = v'_3x^3 + v'_2x^2 + v'_1x + v'_0}$ $z_0 = u'_0 - u_0$, $z_1 = u'_1 - u_1$, $\underline{w}_0 = z_0 \cdot s_0$, $\underline{w}_1 = z_1 \cdot s_1$. Adapted: $\widetilde{t}_2 = -h_2, \widetilde{t}_3 = -h_3$. Reduced: $\widetilde{t}_2 = v_2 + y_2$, $\widetilde{t}_3 = v_3 + y_3$. $t = \widetilde{t}_2 - \widetilde{t}_3u'_1 - \underline{w}_1$. Adapted: $v'_3 = v'_2 = 0$. Reduced: $v'_3 = y_3 + h_3$, $v'_2 = y_2 + h_2$. $v'_1 = (s_0 + s_1) \cdot (z_0 + z_1) - \underline{w}_0 - \underline{w}_1 + u'_1 \cdot t + \widetilde{t}_3u'_0 - v_1 + h_1$, $v'_0 = \underline{w}_0 + u'_0 \cdot t - v_0 + h_0$.	5M	5M
Total		I, 6S, 26M (I, 4S, 28M)	I, 3S, 29M (I, 2S, 29M)

Note: for adapted basis, $v_3 = v_2 = 0$.

Table 3.4: Divisor Doubling Special Case: Adapted Basis

Doubling Special Case, Adapted Basis

Step	Expression	odd	even
$\surd$	Set $r_2 = r^2$. Check $s_1'^2 - s'_1h_3r - f_6r_2 = 0$.	S	S, M
$5'$	$\underline{\text{Inversion}, r^{-1}, s_1, s_0, \widetilde{u}_1^{-1}}$ $\widehat{w}_0 = (2s'_0 + (u_1h_3 - h_2)r)s'_1 - h_3s'_0r - k'_3r_2$,	I, 5M	I, 6M

Continued on next page

Table 3.4 – continued from previous page

Step	Expression	odd	even
	$\widehat{w}_1 = (r\widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0\widehat{w}_1(= \frac{1}{r})$, $\widehat{w}_3 = r_2 r\widehat{w}_1(= \frac{1}{\widetilde{u}_1})$ $s_1 = \widehat{w}_2 s'_1$, $s_0 = \widehat{w}_2 s'_0$		
$6'$	$\underline{u' = x + u'_0}$ $u'_0 = \widehat{w}_3(s_1\widetilde{v}_1 + s_0^2 + (u_1h_3 - h_2)s_0 - k'_2$	S, M	3M
$7'$	$\underline{v' = v'_0}$ $v'_0 = (s_0 - u'_0 s_1)(u'_0(u_1 - u'_0) - u_0) + u'_0(v_1 - h_1 + u'_0(h_2 - u'_0 h_3)) + h_1 - v_0$	3M	S, 4M
Total		I, 5S, 20M	I, 5S, 25M

Table 3.5: Divisor Doubling Special Case: Reduced Basis

Doubling Special Case, Reduced Basis			
Step	Expression	if $s'_1 = 0$	$s'_1 = -t_3 r$
$\surd$	Set $r_2 = r^2$. Check $s_1 = 0$ or $-t_3 r$.	S	S
$5'$	$\underline{\text{Inversion, } r^{-1}, s_1, s_0, \widetilde{u}_1^{-1}}$ $\widehat{w}_0 = s_1((-u_1t_3 + t_2)\cdot r + 2s'_0) + s'_0 t_3 - k'_3 r$ $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1(= \frac{1}{r})$, $\widehat{w}_3 = r_2 \cdot \widehat{w}_1(= \frac{1}{\widetilde{u}_1})$ $s_0 = \widehat{w}_2 \cdot s'_0$	I, 4M	I, 5M
$6'$	$\underline{u' = x + u'_0}$ $u'_0 = \widehat{w}_3 \cdot (s_1\widetilde{v}_1 + s_0 \cdot (-u_1t_3 + t_2 + s_0) - k'_2)$	2M	2M
$7'$	$\underline{v' = v'_3x^3 + v'_2x^2 + v'_1x + v'_0}$	S, 3M	S, 3M

Continued on next page

Table 3.5 – continued from previous page

Step	Expression	if $s'_1 = 0$	$s'_1 = -t_3 r$
	$\widetilde{t}_i = y_i + v_i, i = 1, 2, 3$ $v'_3 = y_3 + h_3$, $v'_2 = y_2 + h_2$, $v'_1 = y_1 + h_1$, $v'_0 = u'_0 \cdot (u'_0 \cdot (u'_0 \widetilde{t}_3 - \widetilde{t}_2) + \widetilde{t}_1) + (u'_0 \cdot (u'_0 - u_1) + u_0) \cdot (s_1 u'_0 - s_0) + h_0 - v_0$		
Total		I, 4S, 21M (I, 3S, 21M)	I, 4S, 22M (I, 3S, 22M)

The general explicit formulas can be used regardless of the field characteristic and the choice of basis (i.e., adapted or reduced). However, in practice, we do not follow the general formulas literally. Instead, assuming isomorphic transformations as introduced in Section 3.3, we rewrite the explicit formulas with respect to different bases and characteristics, so that they are efficient and ready to implement. The results are presented in the following sections 3.6.2.3 and 3.6.2.4.

3.6.2.3 Doubling Formulas in Adapted Basis

Let $[u, v] = [x^2 + u_1 x + u_0, v_1 x + v_0]$ be a degree two divisor in adapted basis with both points of the divisor not equal to their opposites.

We assume the isomorphic transformations in Section 3.3 apply so that the hyperelliptic curve is given in the short Weierstraß form, and count the number of operations accordingly. The resulting explicit formulas are presented in Tables 3.6, 3.7 and 3.8. In Table 3.6, “Odd” and “Even” refer to the parity of the characteristic of K.

Table 3.6: Explicit Formulas for Doubling Divisor Classes in Adapted Basis

Doubling, Adapted Basis, $\deg u = 2$	
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_1 x + v_0$
Output	$[u', v'] = 2[u, v] := [u, v] + [u, v]$

Step	Expression	# of Operations
1	$\underline{\widetilde{v} = \widetilde{v}_1 x + \widetilde{v}_0,\ w_1 = u_1^2.}$ Odd: $\widetilde{v}_1 = 2v_1, \widetilde{v}_0 = u_0 + 2v_0$. Even: $\widetilde{v}_1 = w_1 + u_0 + h_1, \widetilde{v}_0 = u_0 \cdot u_1 + h_0$.	S (S, M)
2	$\underline{r = \mathit{res}(\widetilde{v}, u),\ \mathit{inv} = \mathit{inv}_1 x + \mathit{inv}_0}$ $w_2 = u_0 \cdot \widetilde{v}_1,\ w_3 = u_1 \cdot \widetilde{v}_1$ $\mathit{inv}_1 = -\widetilde{v}_1,\ \mathit{inv}_0 = \widetilde{v}_0 - w_3$ $r = \widetilde{v}_0 \cdot \mathit{inv}_0 + \widetilde{v}_1 \cdot w_2$	4M
3	$\underline{k' \equiv (f + hv - v^2)/u \pmod{u} = k'_1 x + k'_0:}$ Odd: $k'_3 = -2u_1, k'_2 = f_4 + 3w_1 - 2u_0, k'_1 = f_3 + 2(w_1 + u_0 - k'_2) \cdot u_1$, $k'_0 = f_2 - v_1^2 - k'_1 \cdot u_1 - (k'_2 - 4u_0) \cdot (w_1 + 2u_0) - 9u_0^2$. Even: $k'_2 = v_1 + w_1, k'_1 = v_0, k'_0 = f_2 + v_1 \cdot (v_1 + h_1 + w_1) + v_0 \cdot u_1 + w_1^2 + u_0^2$	2S, 3M (2S, 2M)
4	$\underline{s' = s'_1 x + s'_0}$ $s'_1 = \widetilde{v}_0 \cdot k'_1 - \widetilde{v}_1 \cdot k'_0,\ s'_0 = \mathit{inv}_0 \cdot k'_0 + w_2 \cdot k'_1$	4M
$\surd$	Set $r_2 = r^2$. Odd: if $s'_1 = \pm r$, see Table 3.7. Even: if $s'_1 \cdot (s'_1 + r) + r_1 = 0$, see Table 3.8	S (S, M)
5	$\underline{\text{Inversion, } r^{-1},\ s_0,\ s_1,\ \widetilde{u}_2^{-1}}$	I, S, 6M

Continued on next page

Table 3.6 – continued from previous page

Step	Expression	# of Operations
	Odd: $\widehat{w}_0 = s_1'^2 - r_2$. Even: $\widehat{w}_0 = s_1' \cdot (s_1' + r) + f_6 r_2$. $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r \cdot r_2 \cdot \widehat{w}_1 (= \frac{1}{\widetilde{u}_2})$, $s_1 = \widehat{w}_2 \cdot s_1'$, $s_0 = \widehat{w}_2 \cdot s_0'$.	(I, 6M)
6	$\underline{u' = x^2 + u_1'x + u_0'}$ Odd: $u_1' = 2\widehat{w}_3 \cdot (u_1 + s_1 \cdot s_0), u_0' = \widehat{w}_3 \cdot (-k_2' + s_1 \cdot \widetilde{v}_1 + s_0^2)$. Even: $u_1' = \widehat{w}_3 \cdot (s_1 \cdot u_1 + s_0), u_0' = \widehat{w}_3 \cdot (k_2' + s_1 \cdot \widetilde{v}_1 + s_0 \cdot (s_0 + u_1))$.	S, 4M (5M)
7	$\underline{v' = v_1'x + v_0'}$ $z_0 = u_0' - u_0$, $z_1 = u_1' - u_1$, $\underline{w}_0 = z_0 \cdot s_0$, $\underline{w}_1 = z_1 \cdot s_1$. Odd: $v_1' = (z_0 + z_1) \cdot (s_0 + s_1) - \underline{w}_0 - \underline{w}_1 - u_1' \cdot \underline{w}_1 - v_1, v_0' = \underline{w}_0 - u_0' \cdot \underline{w}_1 - v_0$. Even: $v_1' = (z_0 + z_1) \cdot (s_0 + s_1) + \underline{w}_0 + \underline{w}_1 + u_1' \cdot (u_1' + \underline{w}_1) + h_1 + v_1 + u_0'$, $v_0' = \underline{w}_0 + u_0' \cdot (u_1' + \underline{w}_1) + h_0 + v_0$.	5M
Total		I,6S,26M (I, 4S, 28M)

Table 3.7: Divisor Doubling Special Case: Adapted Basis, Odd Characteristic

Doubling Special Case, Adapted Basis, Odd Characteristic

Step	Expression	# of Operations
√	$r_2 = r^2$. Check $s_1' = \pm r$.	S

Continued on next page

Table 3.7 – continued from previous page

Step	Expression	# of Operations
$5'$	$\underline{s = s_1 x + s_0}$ $s_1 = \pm 1$, $\widehat{w}_0 = 2(r \cdot u_1 + s'_0 s_1)$, $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r_2 \cdot \widehat{w}_1 (= \widetilde{u}_1^{-1})$, $s_0 = \widehat{w}_2 \cdot s'_0$	I, 5M
$6'$	$\underline{u' = x + u'_0}$ $u'_0 = \widehat{w}_3 \cdot (2 v_1 s_1 + s_0^2 - k'_2)$	S, M
$7'$	$\underline{v' = v'_0}$ $v'_0 = (s_0 - u'_0 s_1) \cdot (u'_0 \cdot (u_1 - u'_0) - u_0) + u'_0 \cdot v_1 - v_0$	3M
Total		I, 5S, 20M

Table 3.8: Divisor Doubling Special Case: Adapted Basis, Even Characteristic

Doubling Special Case, Adapted Basis, Even Characteristic

Step	Expression	# of Operations
$\surd$	$r_2 = r^2$. Check $s'_1 \cdot (s'_1 + r) + r_2 = 0$.	S, M
$5'$	$\underline{s = s_1 x + s_0}$ $\widehat{w}_0 = u_1 \cdot s'_1 + s'_0$, $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r_2 \cdot \widehat{w}_1 (= \widetilde{u}_1^{-1})$, $s_1 = \widehat{w}_2 \cdot s'_1$, $s_0 = \widehat{w}_2 \cdot s'_0$	I, 6M
$6'$	$\underline{u' = x + u'_0}$ $u'_0 = \widehat{w}_3 \cdot (\widetilde{v}_1 \cdot s_1 + (s_0 + u_1) \cdot s_0 + k'_2)$	3M
$7'$	$\underline{v' = v'_0}$ $w = u_0'^2$, $v'_0 = (s_0 + u'_0 \cdot s_1) \cdot (u'_0 \cdot u_1 + w + u_0) + u'_0 \cdot (v_1 + h_1 + w) + h_0 + v_0$	S, 4M

Continued on next page

Table 3.8 – continued from previous page

Step	Expression	# of Operations
Total		I, 5S, 25M

3.6.2.4 Doubling Formulas in Reduced Basis

Let $[u, v] = [x^2 + u_1 x + u_0, v_3 x^3 + v_2 x^2 + v_1 x + v_0]$ be a degree two divisor in reduced basis with both points of the divisor not equal to their opposites.

Again, we assume the isomorphic transformations in Section 3.3 apply so that the hyperelliptic curve is given in the short Weierstraß form, and count the number of operations accordingly. The resulting explicit formulas are presented in Tables 3.9 and 3.10.

Table 3.9: Explicit Formulas for Doubling Divisor Classes in Reduced Basis

Doubling, Reduced Basis, $\deg u = 2$	
Input	$[u, v], u = x^2 + u_1 x + u_0, v = v_3 x^3 + v_2 x^2 + v_1 x + v_0$
Output	$[u', v'] = 2[u, v] := [u, v] + [u, v]$

Step	Expression	# of Operations
1	$\underline{\widetilde{v} = \widetilde{v}_1 x + \widetilde{v}_0}$ $w_1 = u_1^2$. Odd: $\widetilde{v}_1 = 2(w_1 - u_0 + v_1), \widetilde{v}_0 = 2(u_0 \cdot u_1 + v_0)$. Even: $\widetilde{v}_1 = w_1 + u_0 + h_1, \widetilde{v}_0 = u_0 \cdot u_1 + h_0$.	S, M
2	$\underline{r = res(\widetilde{v}, u),\ inv = inv_1 x + inv_0}$ $w_2 = u_0 \cdot \widetilde{v}_1,\ w_3 = u_1 \cdot \widetilde{v}_1,$	4M

Continued on next page

Table 3.9 – continued from previous page

Step	Expression	# of Operations
	$inv_1 = -\widetilde{v}_1$, $inv_0 = \widetilde{v}_0 - w_3$, $r = \widetilde{v}_0 \cdot inv_0 + \widetilde{v}_1 \cdot w_2$.	
3	$\underline{k' \equiv (f + hv - v^2)/u \pmod{u} = k'_1 x + k'_0}$: Odd: $k'_2 = f_4 - 2v_1, k'_1 = f_3 - 2v_0 - 2u_1 \cdot k'_2, k'_0 = f_2 - v_1^2 - k'_1 \cdot u_1 - k'_2 \cdot (w_1 + 2u_0)$. Even: $k'_2 = h_1 v_3 + v_1, k'_1 = h_0 v_3 + v_0, k'_0 = v_1 \cdot (v_1 + h_1) + f_2 + k'_1 \cdot u_1 + k'_2 \cdot w_1$.	S,3M (3M)
4	$\underline{s' = s'_1 x + s'_0}$ $s'_1 = \widetilde{v}_0 \cdot k'_1 - \widetilde{v}_1 \cdot k'_0$, $s'_0 = w_2 \cdot k'_1 + inv_0 \cdot k'_0$.	4M
$\surd$	If $s'_1 = 0$ or $s'_1 = -t_3 r$, see Table 3.10: Doubling Special Case.	
5	$\underline{\text{Inversion, } r^{-1},\ s_0,\ s_1,\ \widetilde{u}_2^{-1}}$ $r_2 = r^2$. Odd: $\widehat{w}_0 = s'_1 \cdot (2r + s'_1)$. Even: $\widehat{w}_0 = s'_1 \cdot (r + s'_1)$. $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$ $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r \cdot r_2 \cdot \widehat{w}_1 (= \frac{1}{\widetilde{u}_2})$, $s_1 = \widehat{w}_2 \cdot s'_1$, $s_0 = \widehat{w}_2 \cdot s'_0$.	I,S,7M
6	$\underline{u' = x^2 + u'_1 x + u'_0}$ Odd: $u'_1 = 2\widehat{w}_3 \cdot (s_1 \cdot (-u_1 + s_0) + s_0), u'_0 = \widehat{w}_3 \cdot (s_1 \cdot \widetilde{v}_1 + s_0 \cdot (-2u_1 + s_0) - k'_2)$. Even: $u'_1 = \widehat{w}_3 \cdot (s_1 \cdot u_1 + s_0), u'_0 = \widehat{w}_3 \cdot (s_1 \cdot \widetilde{v}_1 + s_0 \cdot (u_1 + s_0) + k'_2)$.	5M
7	$\underline{v' = v'_3 x^3 + v'_2 x^2 + v'_1 x + v'_0}$ $z_0 = u'_0 - u_0$, $z_1 = u'_1 - u_1$, $\underline{w}_0 = z_0 \cdot s_0$, $\underline{w}_1 = z_1 \cdot s_1$. Odd: $t = 2u'_1 + \underline{w}_1, v'_3 = 1, v'_2 = 0, v'_1 = (z_0 + z_1) \cdot (s_0 + s_1) - \underline{w}_0 - \underline{w}_1 - u'_1 \cdot t + 2u'_0 - v_1, v'_0 = \underline{w}_0 - u'_0 \cdot t - v_0$.	5M

Continued on next page

Table 3.9 – continued from previous page

Step	Expression	# of Operations
	Even: $t = u_1' + \underline{w}_1, v_3' = y_3 + h_3, v_2' = y_2 + h_2, v_1' = (z_0 + z_1) \cdot (s_0 + s_1) + \underline{w}_0 + \underline{w}_1 + u_1' \cdot t + u_0' + v_1 + h_1, v_0' = \underline{w}_0 + u_0' \cdot t + v_0 + h_0$.	
Total		I,3S,29M (I,2S,29M)

Table 3.10: Divisor Doubling Special Case: Reduced Basis

Doubling Special Case, Reduced Basis			
Step	Expression	if $s_1' = 0$	$s_1' = -t_3 r$
$5'$	$\underline{\text{Inversion, } r^{-1},\ s_1,\ s_0,\ \widetilde{u}_1^{-1}}$ $t_3 = 2v_3 - h_3, t_2 = 2v_2 - h_2$ $s_1 = 0$ or $-t_3 r$, $\widehat{w}_0 = s_1((-u_1 t_3 + t_2) \cdot r + 2s_0') + s_0' t_3 - k_3' r$ $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$, $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r^2 \cdot \widehat{w}_1 (= \frac{1}{\widetilde{u}_1})$ $s_0 = \widehat{w}_2 \cdot s_0'$	I, S, 4M	I, S, 5M
$6'$	$\underline{u' = x + u_0'}$ $u_0' = \widehat{w}_3 \cdot (s_1 \widetilde{v}_1 + s_0 \cdot (-u_1 t_3 + s_0) - k_2')$.	2M	2M
$7'$	$\underline{v' = v_3' x^3 + v_2' x^2 + v_1' x + v_0'}$ $\widetilde{t}_i = y_i + v_i, i = 1, 2, 3$ $v_3' = y_3 + h_3$, $v_2' = 0$, $v_1' = y_1 + h_1$,	S, 3M	S, 3M

Continued on next page

Table 3.10 – continued from previous page

Step	Expression	if $s'_1 = 0$	$s'_1 = -t_3 r$
	$v'_0 = u'_0 \cdot (u'^2_0 \widetilde{t}_3 + \widetilde{t}_1) + (u'_0 \cdot (u_0 - u_1) + u_0) \cdot (s_1 u'_0 - s_0) + h_0 - v_0$		
Total		I,4S,21M (I,3S,21M)	I,4S,22M (I, 3S, 22M)

Note: In Step 5′, let $\widehat{w}_0 = t_3 s'_0$ if we are in the case $s'_1 = 0$, and let $\widehat{w}_0 = t_3^2 u_1 \cdot r - t_3 s'_0$ if we are in the case $s'_1 = -t_3 r$.

The correctness of the explicit doubling formulas were checked with the computer algebra system MAGMA [53]. We generated test cases, and compared the result of divisor doubling using the explicit formulas and that using MAGMA's built-in generic divisor addition algorithm. We checked that the output divisors were the same with respect to the same input divisors. Note that the divisor representation $[u, v]$ we use in our explicit formulas corresponds to MAGMA's divisor representation $[u, -v]$. This difference was considered in the test.

3.6.3 Summary of Results

The best known results for the imaginary case are found in [34]. Compared to the imaginary case, the addition/doubling formulas for the real case require more multiplications/squarings than the imaginary case. The baby step operation is the cheapest among all operations, and there is no analogue for this operation in the imaginary case. Table 3.11 summarizes the comparison .

Table 3.11: Comparison of Operation Counts for Explicit Formulas (Reduced Basis in Real Case)

	Imaginary		Real	
	odd	even	odd	even
Baby Step	none	none	I, 2S, 4M	I, S, 5M [3]
Addition	I, 2S, 22M [34]	I, 2S, 22M [34]	I, 2S, 26M	I, S, 27M
Doubling	I, 5S, 22M [34]	I, 5S, 22M [34]	I, 3S, 29M	I, 2S, 29M

Key exchange protocols using imaginary and real genus 2 curves have been implemented over large prime fields with explicit formulas. The numerical results can be found in [35]. From the results shown in Table 6 of [35], we conclude that although a bit slower, the operation (field inversion, multiplication and doubling) counts of the divisor addition and doubling formulas for real genus 2 curves are comparable to that of the imaginary genus 2 curves. In certain scenarios of real hyperelliptic curve cryptography, such as the case of "fixed distance" as described in Section 3 of [49], some giant step operations (additions) can be replaced by computationally cheaper baby step operations, under reasonable heuristics that predict the value change of the distance functions involved in the baby step and the giant step (see Section 3 of [49]). With such potential efficiency introduced by the baby step operation, which is unique for the real case, being considered, the use of real genus 2 hyperelliptic curves in cryptography is promising.

---

[3] This result came from Professor Jacobson, in a private communication.

3.7 Future Work

I mention some future directions of this research as follows:

- Derivation of explicit formulas for real genus 2 hyperelliptic curves in projective and mixed coordinates, with which finite field element inversions can be avoided. Explicit formulas using these coordinates may further reduce computational load by trading more expensive field inversions with cheaper field multiplications and squarings.
- Implementation and standardization of cryptographic protocols using real genus 2 hyperelliptic curves. More suitable for small processor architectures, genus 2 curves have their advantage to attract commercial interests. For adoption of cryptosystems using (imaginary or real) genus 2 curves, security, performance, and implementation cost are all among industrial concerns. There are problems like the bandwidth consumption (point compression techniques), generation of suitable curve parameters, compatibility with other cryptographic components (eg., cryptographic hash algorithms), and so on. Like the case of elliptic curves, industrial and governmental acceptance of genus 2 curve based cryptosystems will require a lot of effort. More careful and exhaustive research is needed to justify their usefulness and adoptability.

4. GENERATING SUITABLE PARAMETERS FOR DISCRETE-LOG BASED CRYPTOGRAPHY WITH POLYNOMIAL PARAMETERIZATION

In this chapter, we propose an improved method that uses polynomial parameterization to find suitable parameters for generating cryptographically strong genus 2 curves via the complex multiplication (CM) method. The proposed method is more efficient than the existing method in that it replaces the need for integer factorization with factorization of polynomials with small integral coefficients, which can be done as precomputation, and evaluation of polynomials. We also analyze the probability of success of the proposed method, based on the Bateman-Horn philosophy.

4.1 Introduction

In order to use the Jacobian variety of a curve over a finite field for discrete logarithm based cryptography, suitable parameters must be chosen. One such parameter is the underlying finite field $\mathbb{F}_q$ over which the curve is defined. Another important parameter is the cardinality N of the $\mathbb{F}_q$-rational Jacobian of the curve. For many implementations of discrete logarithm based cryptographic protocols, $\mathbb{F}_q$ is a prime field, i.e., q is a prime number, and N is prime or "close to" a prime number, to resist the Pohlig-Hellman attack [4] to the discrete logarithm problem.

The genus 2 "points counting" methods choose a random curve equation over a finite field and compute the number of points on the Jacobian of the curve until one that is good for cryptography is found. Examples of such methods are [54–56].

An alternative to point counting is to use the genus 2 Complex Multiplication (CM) algorithm to construct curves with a given number of points on its Jacobian.

Like the case of elliptic curve CM method, the genus 2 CM method is very efficient provided that the class polynomials of the complex multiplication field are computed, and that the finite field order q and the order of the Jacobian of the curve N are suitably selected. The genus 2 CM method is a useful alternative, when the genus 2 point counting methods are still slow; and it is especially important for generating pairing-friendly curves as we will see in Chapter 5. For a history of the genus 2 CM method, the reader can refer to [57]. In brief, the algorithm works as follows: Let K be a primitive quartic CM field (explained in 4.2).

1. Find a prime p such that $\exists \omega \in K$ with $\omega\bar{\omega} = p$, and an integer N depending on p and $\mathcal{O}_K$ which will be the group order of the Jacobian of the genus 2 curve having CM by $\mathcal{O}_K$. Such p and N can be identified by using a method in [58].
2. Compute the Igusa class polynomials $H_i(x), i = 1, 2, 3$ of K. This step can be done using the methods described in [57–59] or [60].
3. Construct a curve C from the a set of roots of $H_i(x)$ over $\mathbb{F}_p$ via the Mestre-Cardona-Quer Algorithm [61,62], and check if the Jacobian of the curve has the desired order until a suitable curve is found.

The algorithms described in the following section take as input a given primitive quartic CM field K, and output good cryptographic parameters p and N for a curve C so that C has CM by K and that $\#Jac_{\mathbb{F}_p}(C) = N$.

4.2 Algorithms

Let $K := \mathbb{Q}(\eta)$, where

$$\eta = \begin{cases} i\sqrt{a + b\sqrt{d}} & \text{if } d \equiv 2, 3 \pmod 4 \\ i\sqrt{a + b\frac{-1+\sqrt{d}}{2}} & \text{if } d \equiv 1 \pmod 4 \end{cases},$$

be a fixed primitive quartic CM field, where $d > 0$ is squarefree and $\mathbb{Q}(\sqrt{d})$ has class number 1. The condition that K is primitive is equivalent to $\Delta > 0$ is not a square, where $\Delta = a^2 - b^2 d$, if $d \equiv 2, 3 \pmod 4$, and $\Delta = a^2 - a \cdot b - b^2 \left(\frac{d-1}{4}\right)$, if $d \equiv 1 \pmod 4$. We want to construct a genus 2 hyperelliptic curve C over a finite field $\mathbb{F}_p$[1] of prime order such that $End(Jac_{\mathbb{F}_p}(C)) \otimes \mathbb{Q} = K$, and $N := \#Jac_{\mathbb{F}_p}(C)$ is "almost prime", meaning that N is a product of a large prime number and a small cofactor.

If such a curve C is found, then there exists a element, called the Frobenius element, $\pi \in End(Jac_{\mathbb{F}_p}(C))$ that satisfies the condition $|\pi| = \sqrt{q}$, where $|\pi|$ is the usual absolute value of the complex number π.

Assume for simplicity that the Frobenius element π is in an order

$$\mathcal{O} := \begin{cases} \mathbb{Z} + \sqrt{d}\mathbb{Z} + \eta\mathbb{Z} + \eta\sqrt{d}\mathbb{Z} & \text{if } d \equiv 2, 3 \pmod 4 \\ \mathbb{Z} + \frac{-1+\sqrt{d}}{2}\mathbb{Z} + \eta\mathbb{Z} + \eta\frac{-1+\sqrt{d}}{2}\mathbb{Z} & \text{if } d \equiv 1 \pmod 4 \end{cases}.$$

We first look at the case $d \equiv 2, 3 \pmod 4$ and write $\pi = c_1 + c_2\sqrt{d} + \eta(c_3 + c_4\sqrt{d})$, $c_i \in \mathbb{Z}$.

The relationship $\pi\bar{\pi} = p$ gives us

$$(c_1^2 + c_2^2 d + c_3^2 a + c_4^2 ad + 2c_3c_4bd) + (2c_1c_2 + 2c_3c_4a + c_3^2 b + c_4^2 bd)\sqrt{d} = p.$$

Since 1 and $\sqrt{d}$ are linearly independent over $\mathbb{Q}$ we must have

$$c_1^2 + c_2^2 d + c_3^2 a + c_4^2 ad + 2c_3c_4bd = p \tag{4.1}$$

$$2c_1c_2 + 2c_3c_4a + c_3^2 b + c_4^2 bd = 0 \tag{4.2}$$

Let $\bar{\alpha}$ and α^σ denote the imaginary and real embeddings of K into $\overline{K}$. The characteristic polynomial of π is

$$\begin{aligned} h(x) &= (x - \pi)(x - \bar{\pi})(x - \pi^\sigma)(x - \bar{\pi}^\sigma) \\ &= x^4 - 4c_1x^3 + (2p + 4(c_1^2 - c_2^2 d))x^2 - 4c_1px + p^2 \end{aligned}$$

The fact that $\#Jac_{\mathbb{F}_q}(C) = h(1)$ gives the condition

$$N = (p+1)^2 - 4(p+1)c_1 + 4(c_1^2 - c_2^2 d). \tag{4.3}$$

---

[1] In general, the Mestre's algorithm generates a real model of the genus 2 curve: $y^2 = f(x), \deg(f) = 6$. This real model can be transformed to an isomorphic imaginary model if and only if $f(x)$ has a zero in $\mathbb{F}_p$ [58].

We want N to be almost prime, i.e., $N = c{\cdot}r$ with r prime and c small (say, $c < 2000$).

We have $p \sim N^{\frac{1}{2}}$. Based on the discussions above, A. Weng (see [58]) gives a probabilistic method of searching for parameters, which produces prime p and almost prime N. In this method, factorization of big integers is used repeatedly in every step of the search[2], which makes the algorithm slow.

We give a more efficient algorithm which generates parameters for genus 2 cryptography. The idea is to parameterize the coefficients c_i's as polynomials $c_i(x)$, then generate "families of parameters" by factorizing quartic polynomials with small integral coefficients.

To this end, we try to find polynomials $c_1(x), c_2(x), c_3(x), c_4(x) \in \mathbb{Q}[x]$ satisfying $-2c_1(x)c_2(x) = 2c_3(x)c_4(x)a + c_3^2(x)b + c_4^2(x)bd$. Then we write $p(x) = c_1^2(x) + c_2^2(x)d + c_3^2(x)a + c_4^2(x)ad + 2c_3(x)c_4(x)bd$ and let x range through integer values of certain sizes until the value $p(x)$ is a prime number. Now we can use Equation (4.3) to compute the cardinality of the Jacobian and check if it is almost prime.

The following lemma helps us avoid some bad choices of $c_i(x)$.

Lemma 12 *Let $c_1(x), c_2(x), c_3(x), c_4(x)$ be linear polynomials in $\mathbb{Q}[x]$ such that*

$$2c_1(x)c_2(x) + 2c_3(x)c_4(x)a + c_3^2(x)b + c_4^2(x)bd = 0.$$

Then

$$p(x) = c_1^2(x) + c_2^2(x)d + c_3^2(x)a + c_4^2(x)ad + 2c_3(x)c_4(x)bd$$

is reducible in $\mathbb{Q}[x]$.

Proof Let $c_1(x), c_2(x), c_3(x), c_4(x)$ be linear polynomials in $\mathbb{Q}[x]$ such that $2c_1(x)c_2(x) + 2c_3(x)c_4(x)a + c_3^2(x)b + c_4^2(x)bd = 0$. Then we have

$$-2c_1(x)c_2(x) = 2c_3(x)c_4(x)a + c_3^2(x)b + c_4^2(x)bd. \tag{4.4}$$

Let $\alpha \in \mathbb{Q}$ be a root of $c_1(x)$. Clearly,

$$0 = -2c_1(\alpha)c_2(\alpha) = bc_3^2(\alpha) + 2ac_3(\alpha)c_4(\alpha) + bdc_4^2(\alpha) = 0. \tag{4.5}$$

---

[2] This method chooses random c_3 and c_4 in Equation (4.2), and factors $2c_3c_4a + c_3^2b + c_4bd$ to obtain possible choices of c_1 and c_2.

Now we look at the quadratic equations

$$bX^2 + 2aX + bd = 0 \tag{4.6}$$

$$bdX^2 + 2aX + b = 0. \tag{4.7}$$

Both equations (4.6) and (4.7) have discriminant $\Delta = (2a)^2 - 4b(bd) = 4(a^2 - b^2 d) > 0$, which is not a square in $\mathbb{Q}$ by the assumption on a, b and d, namely, that K is primitive. Therefore Equation (4.5) holds if and only if $c_3(\alpha) = c_4(\alpha) = 0$. Hence $c_3(\alpha) = c_4(\alpha) = 0$. By Equation (4.5) we conclude that α is a zero of $-2c_1(x)c_2(x)$ with multiplicity 2. Since $c_1(x)$ and $c_2(x)$ are linear, we must have $c_2(\alpha) = 0$.

Therefore, the polynomial

$$p(x) = c_1^2(x) + c_2^2(x)d + c_3^2(x)a + c_4^2(x)ad + 2c_3(x)c_4(x)bd$$

has α as a zero of multiplicity 2. So $(x-\alpha)^2 | p(x)$ in $\mathbb{Q}[x]$. Obviously, $p(x)$ is reducible.

■

Since we want $p(x)$ to be prime for some values $x \in \mathbb{Z}$, we expect $p(x)$ to be irreducible and have no fixed prime divisors. Here we say that a prime number $\mathfrak{p}$ is a *fixed prime divisor* of a polynomial $f(x)$ with rational coefficients if $\mathfrak{p}$ divides every integer-valued $f(n)$ for $n \in \mathbb{Z}$. We also define the *greatest fixed divisor* $GFD(f)$ to be the largest positive integer d such that d divides all integral values of $f(n)$ for $n \in \mathbb{Z}$. We say that a polynomial $f(x)$ with **integral coefficients** has the *Bunyakovsky's property* if $f(x)$ has no fixed prime divisors. We need to check that the irreducible $p(x)$ has no fixed prime divisors, i.e., it satisfies Bunyakovsky's property. This can be easily checked by using Newton's interpolation formula (see, e.g., [63], Section 2.2) to write $p(x)$ in terms of polynomials basis $\{x(x-1)(x-2)\dots(x-k+1)/k!\}_{k\in\mathbb{Z}^+}$, then verifying that the coefficients are relatively prime [64].

Lemma 12 implies that we cannot choose linear polynomials $c_3(x)$ and $c_4(x)$ to obtain such $p(x)$. Therefore, we choose $c_3(x)$ and $c_4(x)$ to be quadratic polynomials in the following algorithm[3].

---

[3] Alternatively, we can choose one of $c_3(x)$ and $c_4(x)$ to be a quadratic polynomial and the other is a linear polynomial or a constant. According to the discussion in Section 4.3, the performance of the proposed algorithms will not be affected much with such alternatives.

Algorithm 4 Parameter generator polynomials for $K = \mathbb{Q}(\eta)$, $d \equiv 2, 3 \pmod 4$

Input: Integers a, b, d with $d > 0$ squarefree, $d \equiv 2, 3 \pmod 4$, $a^2 - b^2d > 0$ not a square.

Output: Four quadratic polynomials $c_1(x), c_2(x), c_3(x), c_4(x)$ and a quartic polynomial $p(x)$ are generated such that they satisfy the equations (4.1) and (4.2). Polynomials $N_1(x)$ and $N_2(x)$ of degree 8 are generated as possible group orders.

```
repeat
    repeat
        Choose quadratic polynomials $c_3(x)$ and $c_4(x)$ in $\mathbb{Z}[x]$ with small coefficients and $\gcd(c_3(x), c_4(x)) = 1$.
        Set $n(x) = 2c_3(x)c_4(x)a + c_3^2(x)b + c_4^2(x)bd$.
    until $\deg n(x) = 4$ and $n(x) = \widetilde{c_1}(x) \cdot \widetilde{c_2}(x)$, $\deg \widetilde{c_1}(x) = 2 = \deg \widetilde{c_2}(x)$, $\gcd(\widetilde{c_1}(x), \widetilde{c_2}(x)) = 1$, $n(x)$ and $\widetilde{c_1}(x)$ have the same content.
    Set $c_1(x) = -\frac{1}{2}\widetilde{c_1}(x)$, $c_2(x) = \widetilde{c_2}(x)$.
    Set $p(x) = c_1^2(x) + c_2^2(x)d + c_3^2(x)a + c_4^2(x)ad + 2c_3(x)c_4(x)bd$.
until $p(x)$ is irreducible and has no fixed prime divisor.
Set $N_1(x) = (p(x)+1)^2 - 4(p(x)+1)c_1(x) + 4(c_1^2(x) - c_2^2(x)d)$,
    $N_2(x) = (p(x)+1)^2 + 4(p(x)+1)c_1(x) + 4(c_1^2(x) - c_2^2(x)d)$.
```

We have a similar result for the case $d \equiv 1 \pmod 4$.

In this case, we write

$$\pi = c_1 + c_2\frac{-1+\sqrt{d}}{2} + \eta\left(c_3 + c_4\frac{-1+\sqrt{d}}{2}\right), c_i \in \mathbb{Z}.$$

Again, $\pi\bar{\pi} = p$ gives

$$\left(c_1^2 + \left(\tfrac{d-1}{4}\right)c_2^2 + ac_3^2 + 2b\left(\tfrac{d-1}{4}\right)c_3c_4 + (a-b)\left(\tfrac{d-1}{4}\right)c_4^2\right)$$
$$+ \left(2c_1c_2 - c_2^2 + bc_3^2 + 2(a-b)c_3c_4 + \left(b\left(\tfrac{d+3}{4}\right) - a\right)c_4^2\right)\tfrac{-1+\sqrt{d}}{2} = p.$$

The linear independence of 1 and $\frac{-1+\sqrt{d}}{2}$ over $\mathbb{Q}$ implies the following two equations

$$\begin{aligned} c_1^2 + \left(\frac{d-1}{4}\right) c_2^2 + a c_3^2 + 2b\left(\frac{d-1}{4}\right) c_3 c_4 + (a-b)\left(\frac{d-1}{4}\right) c_4^2 &= p \qquad (4.8) \\ 2c_1c_2 - c_2^2 + bc_3^2 + 2(a-b)c_3c_4 + \left(b\left(\frac{d+3}{4}\right) - a\right) c_4^2 &= 0 \qquad (4.9) \end{aligned}$$

The corresponding cardinality of the Jacobian

$$N = (p+1)^2 - (4c_1 - 2c_2)(p+1) + 4\left(c_1^2 - c_1c_2 - \left(\frac{d-1}{4}\right)c_2^2\right). \tag{4.10}$$

The algorithm is given as follows.

Algorithm 5 Parameter generator polynomials for $K = \mathbb{Q}(\eta)$, $d \equiv 1 \pmod 4$

Input: Integers a, b, d with $d > 0$ squarefree, $d \equiv 1 \pmod 4$, $a^2 - ab - b^2\left(\frac{d-1}{4}\right) > 0$ not a square.

Output: Four quadratic polynomials $c_1(x), c_2(x), c_3(x), c_4(x)$ and a quartic polynomial $p(x)$ are generated such that they satisfy the equations (4.8) and (4.9). Polynomials $N_1(x)$ and $N_2(x)$ of degree 8 are generated as possible group orders.

```
repeat
    repeat
        Choose quadratic polynomials $c_3(x)$ and $c_4(x)$ in $\mathbb{Z}[x]$ with small coefficients and $\gcd(c_3(x), c_4(x)) = 1$.
        Set $n(x) = 2c_3(x)c_4(x)a - c_4^2(x)a + c_3^2(x)b - 2c_3(x)c_4(x)b + c_4^2(x)b(\frac{d+3}{4})$.
    until $\deg n(x) = 4$ and $n(x) = \widetilde{c}_1(x) \cdot \widetilde{c}_2(x)$, $\deg \widetilde{c}_1(x) = 2 = \deg \widetilde{c}_2(x)$, $\gcd(\widetilde{c}_1(x), \widetilde{c}_2(x)) = 1$, $n(x)$ and $\widetilde{c}_1(x)$ have the same content.
    Set $c_2(x) = \widetilde{c}_2(x)$, $c_1(x) = \frac{1}{2}\left(-\widetilde{c}_1(x) + c_2(x)\right)$.
    Set $p(x) = c_1^2(x) + c_2^2(x)\left(\frac{d-1}{4}\right) + c_3^2(x)a + c_4^2(x)a\left(\frac{d-1}{4}\right) + 2c_3(x)c_4(x)b\left(\frac{d-1}{4}\right) - bc_4^2(x)\left(\frac{d-1}{4}\right)$.
until $p(x)$ is irreducible and has no fixed prime divisor.
Set $N_1(x) = (p(x)+1)^2 - (p(x)+1)(4c_1(x) - 2c_2(x)) + 4\left(c_1^2(x) - c_1(x)c_2(x) - c_2^2(x)\left(\frac{d-1}{4}\right)\right)$,
    $N_2(x) = (p(x)+1)^2 + (p(x)+1)(4c_1(x) - 2c_2(x)) + 4\left(c_1^2(x) - c_1(x)c_2(x) - c_2^2(x)\left(\frac{d-1}{4}\right)\right)$.
```

The polynomials returned from Algorithms 4 and Algorithm 5 are candidates of parameter generator polynomials. We then insert integer values of a suitable size into them until $p(x)$ is prime and $N_1(x)$ or $N_2(x)$ is almost prime. This process is written formally in Algorithm 6 as follows.

Algorithm 6 Algorithm for generating parameters for HEC cryptography

Input: Polynomials $c_1(x), p(x), N_1(x)$ and $N_2(x)$ generated by Algorithm 4 or Algorithm 5; bit length, μ, of the desired size of the prime field over which the curve is defined; maximum number of trials, M.

Output: Triples (p, N_1, N_2) for constructing hyperelliptic curves over $\mathbb{F}_p$ with CM by $K = \mathbb{Q}(\eta)$ whose Jacobians have almost prime group orders N_1 or $N_2 \sim 2^{2\mu}$; or "Not found".

```
number_of_trial = 0.
repeat
    Choose an integer $x_0 \sim 2^{\frac{\mu}{4}}$.
    if $c_1(x_0)$ is an integer then
        $p \leftarrow p(x_0)$.
        if $p$ is prime and $2^{\mu-1} < p < 2^{\mu}$, and either $N_1 \leftarrow N_1(x_0)$ or $N_2 \leftarrow N_2(x_0)$ is almost prime then
            Return $(p, N_1, N_2)$.
        end if
    end if
    number_of_trial ← number_of_trial + 1.
until number_of_trial = $M$.
Return "Not found".
```

In Step 6 of Algorithm 6, to test if $N_i (i = 1, 2)$ is almost prime, we can find the maximum factor $h_{\max}$ of N_i below a specified upper bound (i.e., 10,000), and perform a primality test for $N_i/h_{\max}$. In this way, factorization for large integers can be avoided in our method.

We implemented Algorithms 4, 5 and 6 as well as the Weng's method for generating (prime, group order) pairs (p, N) with respect to randomly chosen quartic CM fields, specified by small parameters a, b, d, randomly chosen so that $0 < d \leq 50$ is squarefree, $\mathbb{Q}(\sqrt{d})$ has class number one, $0 < |a|, |b| \leq 50$, and $\Delta > 0$ is not a square. Trial factorization of integers up to a fixed bound (10,000) is used for the old method.

1. **128-bit p:** In the case $d \equiv 2, 3 \pmod 4$, our method generates parameter pairs (p, N) at an average rate of 2.1402 seconds per pair, while Weng's algorithm generates pairs at 7.7538 seconds per pair. In the case $d \equiv 1 \pmod 4$, our method generates parameter pairs at an average rate of 3.6423 seconds per pair, and Weng's method at 11.4407 seconds per pair.

2. **256-bit p:** In the case $d \equiv 2, 3 \pmod 4$, our method generates parameter pairs at an average rate of 21.7344 seconds per pair, while Weng's algorithm generates pairs at 97.9592 seconds per pair. In the case $d \equiv 1 \pmod 4$, our method generates parameter pairs at an average rate of 41.0917 seconds per pair, and Weng's method at 108.5106 seconds per pair.

However, we notice that there are rare cases in which the algorithm fails to find suitable parameters. We expect our method to perform much better on average as the size of the prime p increases, if complete factorization of integers is used for Weng's method.

The implementation is performed in PARI/GP [65]. We include in Appendix A some parameters found by the above algorithms.

4.3 Probability That $p(x)$ is Prime and $N_i(x)$ is Almost Prime

Although in practice we allow polynomials with rational coefficients, for simplicity of the analysis of Algorithms 6 above, we choose $c_i(x)$ to be polynomials with integral coefficients. We set $N(x)$ to be one of $N_1(x)$ and $N_2(x)$. We also require that $N(x)$ be irreducible over $\mathbb{Q}$. Experiment shows that if $p(x)$ is irreducible then $N(x)$ is also irreducible with very high probability.

We do not intend to provide an exhaustive analysis in this thesis. Rather, we give some intuition about how often it happens that $p(m)$ is prime and $N(m)$ is almost prime simultaneously for an integer m. Our argument is related to the Bateman-Horn Conjecture [66,67] and its generalized version for the case of a single polynomial [64].

The Bateman-Horn Conjecture is a quantitative form of Hypothesis H of A. Schinzel [68, 69]. It approximates the density of the positive integers at which a given set of polynomials have simultaneous integer values. It is stated as follows.

Conjecture 1 (Bateman-Horn, [66]) *Suppose $f_1, f_2, \ldots, f_k$ are polynomials in one variable with all coefficients integral and leading coefficients positive, their degree being $h_1, h_2, \ldots, h_k$ respectively. Suppose each f_i is irreducible over $\mathbb{Q}$ and no two of them differ by a constant factor (i.e., they are not associated). Let $Q(f_1, f_2, \ldots, f_k; M)$ denote the number of positive integers n between 1 and M inclusive such that $f_1(n)$, $f_2(n)$, …, $f_k(n)$ are all primes (ignoring the finitely many values of n for which some $f_i(n)$ is negative). Then we have*

$$Q(f_1, f_2, \ldots, f_k; M) \sim h^{-1} C \int_2^M (\log u)^{-k} \, du,$$

where

$$C = \prod_{\mathfrak{p} \text{ prime}} \left(1 - \frac{1}{\mathfrak{p}}\right)^{-k} \left(1 - \frac{W(\mathfrak{p})}{\mathfrak{p}}\right), \quad h = h_1^{-1} h_2^{-1} \ldots h_k^{-1},$$

and $W(\mathfrak{p})$ is the number of solutions of the congruence

$$f_1(x) f_2(x) \ldots f_k(x) \equiv 0 \pmod{\mathfrak{p}}.$$

For the case of one polynomial f, the Bateman-Horn Conjecture is generalized by M. Adleman and A. Odlyzko to deal with the situation $GFD(f) \neq 1$. We summarize it as follows.

Conjecture 2 (Adleman-Odlyzko, [64]) *Suppose $f(x) \in \mathbb{Z}[x]$ is irreducible with greatest fixed divisor $d = GFD(f)$ and has a positive leading coefficient. For a prime $\mathfrak{p}$, let $r = r_{\mathfrak{p}}$ be the least nonnegative integer such that the values of $f(m)d^{-1}$, when* *reduced modulo* $\mathfrak{p}$*, are periodic in* m *with period* $\mathfrak{p}^{r+1}$*. (It can be shown that* $r \leq \frac{\deg(f)}{\mathfrak{p}-1}$*.) Let*

$$W(\mathfrak{p}^{r+1}) = \#\left\{m : 0 \leq m < \mathfrak{p}^{r+1}, f(m) \equiv 0 \pmod{\mathfrak{p}}\right\}.$$

Let $Q(f; M) = \#\{m : 1 \leq m \leq M, f(m)d^{-1}$ *is a prime*$\}$*. Then*

$$Q(f; M) \sim \frac{M}{h \log(M)} C,$$

where $h = \deg(f)$ *and*

$$C = \prod_{\mathfrak{p} \textit{ prime}} \left(1 - \frac{1}{\mathfrak{p}}\right)^{-1} \left(1 - \frac{W(\mathfrak{p}^{r+1})}{\mathfrak{p}^{r+1}}\right).$$

Based on the above two conjectures, we conjecture, for analysis of our algorithm, as follows.

Conjecture 3 *Let* $p(x) \in \mathbb{Z}[x]$ *be irreducible with no fixed prime divisor and a positive leading coefficient. Let* $N(x) \in \mathbb{Z}[x]$ *be irreducible with greatest fixed divisor* $d = GFD(N)$ *and a positive leading coefficient. Let* $f(x) = p(x) \cdot d^{-1}N(x)$ *and let* $r = r_{\mathfrak{p}}$ *be the least nonnegative integer such that the values of* $f(m)$*, when reduced modulo a prime* $\mathfrak{p}$*, are periodic in* m *with period* $\mathfrak{p}^{r+1}$*. We have the fact that* $r \leq \frac{\deg(f)}{\mathfrak{p}-1}$ *[64]. Let*

$$W(\mathfrak{p}^{r+1}) = \#\left\{m : 0 \leq m < \mathfrak{p}^{r+1}, f(m) \equiv 0 \pmod{\mathfrak{p}}\right\}.$$

Let $Q(p, N; M) = \#\{m : 1 \leq m \leq M, p(m)$ *and* $d^{-1}N(m)$ *are both prime*$\}$*. Then*

$$Q(p, N; M) \sim h^{-1} C \int_2^M (\log u)^{-2}\, du, \tag{4.11}$$

where $h = \deg(p)\deg(N)$ *and*

$$C = \prod_{\mathfrak{p} \textit{ prime}} \left(1 - \frac{1}{\mathfrak{p}}\right)^{-2} \left(1 - \frac{W(\mathfrak{p}^{r+1})}{\mathfrak{p}^{r+1}}\right).$$

Example 1. Let $p(x) = x^2 + 1$ and $N(x) = x^2 + x + 2$ with $d = GFD(N) = 2$. We consider the number of prime pairs $(p(m), d^{-1}N(m))$ for $2^{22} \leq m \leq 2^{23}$. We have $h = 4$, $C \approx 2.6741$ and $\int_{2^{22}}^{2^{23}} \log(u)^{-2}\, du \approx 17165$. Thus the estimated value of $Q_1 := Q(p, N; 2^{23}) - Q(p, N; 2^{22})$ is approximated 11475. On the other hand, an explicit counting shows that $Q_1 = 11844$.

Conjecture 3 suggests a lower bound for the number of pairs $(p(m), N(m))$ with $p(m)$ prime and $N(m)$ almost prime. And it must be noted that this lower bound does not work well for the case of polynomials $p(x)$ and $N(x)$ obtained in Algorithm 4 and 5, because $N(m)$ is always divisible by 4 when $p(m)$ is odd, in particular, prime, and this will lead to $C = 0$. But $N(m)$ is odd when $p(m)$ is even. However, in this situation, it seems natural to consider pairs of primes of the form $(p(m), d^{-1}N(x)/4)$. We claim that an estimate in the form of (4.11) still holds, but the computation of the constant C needs to be modified accordingly, by considering the case $\mathfrak{p} = 2$ separately. We will show a way to do this in the following example.

Example 2. Let $p(x) = x^2+1$ and $N(x) = (p(x)+1)^2+4(p(x)+1)+8 = x^4+8x^2+20$. Then neither $p(x)$ nor $N(x)$ have a fixed prime divisor and $N(m)$ has 4 as a divisor if $p(m)$ is odd. Let $f(x) = p(x) \cdot N(x)$ and $W(\mathfrak{p})$ be the number of solutions for the equation $f(m) \equiv 0 \pmod{\mathfrak{p}}$. If we let $Q(M)$ be the number of prime pairs $(p(m), N(m)/4)$ for $1 \le m \le M$, then we claim that $Q(M)$ can be estimated as

$$Q(M) \sim h^{-1} C \int_2^M (\log u)^{-2}\, du, \tag{4.12}$$

where $h = 2 \cdot 4 = 8$ and

$$C = C_2 \cdot \prod_{\mathfrak{p} \geq 3 \text{ prime}} \left(1 - \frac{1}{\mathfrak{p}}\right)^{-2} \left(1 - \frac{W(\mathfrak{p})}{\mathfrak{p}}\right) \approx C_2 \cdot 2.3775,$$

where C_2 is computed, based on the philosophy of Bateman-Horn, as

$$\begin{aligned} C_2 &= \frac{\mathit{Prob}\,\{N(m)/4 \in \mathbb{Z} \text{ and both } p(m) \text{ and } N(m)/4 \text{ are not divisible by } 2\}}{\mathit{Prob}\,\{\text{two random numbers are not divisible by } 2\}} \\ &= \frac{\mathit{Prob}\,\{p(m) \text{ is odd}\} \cdot \mathit{Prob}\,\{N(m)/4 \text{ is odd} \mid p(m) \text{ is odd}\}}{(1-1/2)^2} \\ &= 4 \cdot \mathit{Prob}\,\{m \text{ is even}\}\, \mathit{Prob}\,\{N(m)/4 \text{ is odd} \mid m \text{ is even}\} \\ &= 4 \cdot \frac{1}{2} \cdot 1 = 2. \end{aligned}$$

Therefore, we have $Q := Q(2^{23}) - Q(2^{22}) \approx 10202$ by (4.12). A direct counting gives $Q = 10483$, close to the estimate.

We want to keep the degrees of $p(x)$ and $N(x)$ as small as possible so that the probability of obtaining suitable parameters of a fixed size is as high as possible, as indicated by the above heuristic. Our algorithms produce $p(x)$ of degree 4 and $N_i(x)$ of degree 8.

Note that $c_3(x)$ and $c_4(x)$ can also be chosen in such a way that one is a linear polynomial and the other is a quadratic polynomial. In this case, the resulting $p(x)$ and $N_i(x)$ are still of degree 4 and 8, respectively. Therefore the probability of getting suitable pairs $(p(m), N(m))$ should not differ significantly.

4.4 Conclusion and Further Discussion

We present in this chapter a method of generating cryptographically strong parameters for constructing genus 2 hyperelliptic curves. The method shows an improvement over the existing method by using polynomial parameterization. It efficiently generates parameters p and N *approximately* of a certain size by choosing a suitable sized x_0 in Algorithm 6. However, if an *exact* size (of p or N) is specified by the practical requirement, the value of x_0 should be chosen more carefully.

Future research directions include improvement of the algorithms so that they can effectively output parameters of exact lengths, and more efficient implementation. Although it seems hard to prove the conjectures like Bateman-Horn, generalizing Conjecture 3 so that it works for multiple polynomials that do not have the Bunyakovsky's property is another topic of research.

Generating pairing-friendly parameters is another direction of research. We shall discuss this extension in the chapter that immediately follows.

5. GENERATING PARAMETERS FOR PAIRING-FRIENDLY GENUS 2 CURVES OVER PRIME FIELDS

We present two contributions in this chapter.

First, we give a quantitative analysis of the scarcity of pairing-friendly genus 2 curves, assuming the Riemann Hypothesis. This result shows an improvement relative to prior work [70], which does a heuristic estimation of the density of pairing-friendly genus 2 curves.

Second, we present a method for generating pairing-friendly parameters for which $\rho \approx 8$[1] This method applies the idea of [71] to a setting given in terms of coefficients of the Frobenius element. It is easy to understand and implement.

5.1 Introduction

For a (multiplicatively written) cyclic group G of order q, with a generator $g \in G$, the *Decision Diffie-Hellman Problem* is the following problem: Given g^a and g^b for randomly-chosen $a, b \in \{0, \ldots, q-1\}$, and $h \in G$, determine if $c = g^{ab}$; and the *Computational Diffie-Hellman Problem* is the following problem: Given g^a and g^b for randomly-chosen $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

Many cryptographic algorithms and protocols make use of “bilinear maps”, with which there are groups in which the *Decision Diffie-Hellman Problem* is easy and the *Computational Diffie-Hellman Problem* is hard [72]. Such maps have application in identity-based encryption [73], aggregate signatures [72], short signatures [74], tripartite key agreement protocols [75], non-interactive key distribution protocols [76], and

---

[1] The definition of ρ can be found in Section 5.4.

so on. Pairings for elliptic curves and Jacobian varieties of hyperelliptic curves provide efficient implementation of such bilinear maps, which are suitable for cryptography.

In addition to the requirements that must be satisfied for discrete-log based cryptography, pairing-based cryptography poses further restrictions on the curves — for many applications, a low "embedding degree" is desired. The low density of pairing-friendly curves among cryptographically strong ones makes it extremely hard to find suitable curves for pairing-based crypto via point counting. This indicates that the CM method is probably the only suitable method for finding pairing-friendly genus 2 curves nowadays. For the pairing-friendly setting, a bound on the desired embedding degree is an additional input to the CM algorithm.

5.2 Weil and Tate-Lichtenbaum Pairings

For an abelian variety $\mathcal{A}$ over a finite field F and an integer r coprime to the characteristic of F, a Weil pairing is a nondegenerate, skew-symmetric bilinear map

$$e_r^W : \mathcal{A}(\bar{F})[r] \times \mathcal{A}(\bar{F})[r] \to \mu_r(\bar{F}),$$

where $\bar{F}$ is an algebraic closure of F and $\mu_r(\bar{F})$ is the group of r^{th} roots of unity in $\bar{F}$; a Tate-Lichtenbaum pairing is a nondegenerate bilinear map

$$e_r^{TL} : \mathcal{A}(F)[r] \times \mathcal{A}(F)/r\mathcal{A}(F) \to F^*/(F^*)^r.$$

$F^*/(F^*)^r$ is isomorphic to $\mu_r(\bar{F})$ if and only if $\mu_r(\bar{F}) \subseteq F$. In many pairing-based cryptographic applications, we want the target group of a pairing map to have an element of order r, so we need to work over a field containing the r^{th} roots of unity.

Definition 13 (Embedding degree) *Let $\mathcal{A}$ be an abelian variety over a finite field $F = \mathbb{F}_q$. Let r be an integer coprime to q which divides $\#\mathcal{A}(F)$. The field $F(\mu_r(\bar{F}))$ is a finite extension $\mathbb{F}_{q^k}$ of F. The number k is called the* **embedding degree of $\mathcal{A}$ with respect to** *r, and it is the smallest integer such that $r | q^k - 1$.*

We also call the embedding degree of the Jacobian of a nonsingular projective curve C the "embedding degree of the curve C."

In many settings relevant to pairing-based cryptography, we need an abelian variety $\mathcal{A}$ with $\#\mathcal{A}$ almost prime, i.e., $\#\mathcal{A} = h \cdot r$, where h is a small positive integer and r is a prime number; and the embedding degree k of $\mathcal{A}$ with respect to r is not too large.

Definition 14 (Pairing-friendly abelian variety) *Let H and K be positive integers. Let $\mathcal{A}$ be an abelian variety over a finite field $\mathbb{F}_q$. We say $\mathcal{A}$ is* **pairing-friendly with respect to parameters** H **and** K *if $\#\mathcal{A} = h \cdot r$ for some positive integer $h \leq H$ and a prime number r, and the embedding degree k of $\mathcal{A}$ with respect to r is no larger than K.*

By convention, we call an abelian variety "pairing-friendly" if H and K are "small". We also say a nonsingular projective curve C is "pairing-friendly" if C has a pairing-friendly Jacobian.

5.3 Pairing-friendly Genus 2 Curves Are Rare

In this section, we shall show (assuming the Riemann Hypothesis) that there are very few pairing-friendly genus 2 hyperelliptic curves among all genus 2 hyperelliptic curves over prime fields whose Jacobians have almost prime orders. A similar result for elliptic curves is proved in [77].

Let p be an odd prime number, and let $\log(\cdot)$ denote the natural logarithm.

The main result of this section is Theorem 18. Before proving it, we first introduce several lemmas.

Lemma 15 *Let M and c be positive constants with $c < 4$. For a fixed positive integer a, let $\mathcal{S}_{a,c,M}$ denote the set of pairs of primes (x, y) such that $\frac{M}{2} \leq x \leq M$ and*

$|x^2 - a \cdot y| \leq c \cdot x^{3/2}$. *If the Riemann Hypothesis (R.H.) holds, then for large enough* M*, we have*

$$|\mathcal{S}_{a,c,M}| \geq \frac{1}{15} \cdot \frac{c}{a} \cdot \frac{M^{5/2}}{(\log M)^2}.$$

Proof Let $\pi(x)$ be number of primes in the interval $[1, x]$. Let $N = \pi(M) - \pi(\frac{M}{2})$ be the number of primes in $(M/2, M]$. The Prime Number Theorem (P.N.T.) implies $N > \frac{1}{3} \cdot \frac{M}{\log M}$ for M large enough.

Now let p be a prime number in $(M/2, M]$. We look at the number of primes y such that $|p^2 - a \cdot y| \leq c \cdot p^{3/2}$, i.e., $\frac{1}{a}\left(p^2 - c \cdot p^{3/2}\right) \leq y \leq \frac{1}{a}\left(p^2 + c \cdot p^{3/2}\right)$. Denote this number by N_p. By a theorem of von Koch (see [78], Theorem 8.3.3), if the R.H. is true,

$$\pi(x) = \mathrm{li}(x) + O(\sqrt{x}\log x),$$

where $\mathrm{li}(x) = \int_2^x \mathrm{d}t/\log t$. Moreover, by a result of L. Schoenfeld (see [79], Corollary 1), if R.H. is true, there exists an effectively computable positive constant c_1 such that $|\pi(x) - \mathrm{li}(x)| < c_1 \cdot \sqrt{x}\log x$, when $x \geq 2657$. According to this result, when p is large, we have

$$\begin{aligned}
N_p &\geq \pi\left(\frac{1}{a}\left(p^2 + c \cdot p^{3/2}\right)\right) - \pi\left(\frac{1}{a}\left(p^2 - c \cdot p^{3/2}\right)\right) \\
&> \mathrm{li}\left(\frac{1}{a}\left(p^2 + c \cdot p^{3/2}\right)\right) - \mathrm{li}\left(\frac{1}{a}\left(p^2 - c \cdot p^{3/2}\right)\right) \\
&\quad -\frac{1}{a} \cdot c_1(p^2 + c \cdot p^{3/2})^{1/2} \log\left(p^2 + c \cdot p^{3/2}\right) \\
&\quad -\frac{1}{a} \cdot c_1\left(p^2 - c \cdot p^{3/2}\right)^{1/2} \log\left(p^2 - c \cdot p^{3/2}\right) \\
&> \int_{\frac{1}{a}\left(p^2 - c \cdot p^{3/2}\right)}^{\frac{1}{a}\left(p^2 + c \cdot p^{3/2}\right)} \frac{\mathrm{d}t}{\log t} - \frac{1}{a} \cdot 2c_1\left(p^2 + c \cdot p^{3/2}\right)^{1/2} \log\left(p^2 + c \cdot p^{3/2}\right) \\
&> \frac{1}{\log\left(\frac{1}{a} \cdot (M^2 + c \cdot M^{3/2})\right)} \cdot \frac{1}{a}\left(2c\left(\frac{M}{2}\right)^{3/2}\right) - \frac{1}{a} \cdot 2c_1\left(2M^2\right)^{1/2} \log\left(2M^2\right) \\
&> \frac{1}{\log(2M^2) - \log a} \cdot \frac{c}{a\sqrt{2}} M^{3/2} - \frac{1}{a} \cdot 8c_1 M \log M \\
&> \frac{1}{a}\left(\frac{cM^{3/2}}{4\log M} - 8c_1(M\log M)\right) \\
&> \frac{1}{5} \cdot \frac{c}{a} \cdot \frac{M^{3/2}}{\log M}.
\end{aligned}$$

Note that the inequality above is independent of the value p. Summing over all suitable primes p, we obtain

$$|\mathcal{S}_{a,c,M}| = \sum_{\substack{\frac{M}{2} \le p \le M \\ p \text{ prime}}} N_p \ge \frac{1}{5} \cdot \frac{c}{a} \cdot \frac{M^{3/2}}{\log M} \cdot \frac{1}{3} \cdot \frac{M}{\log M} = \frac{1}{15} \cdot \frac{c}{a} \cdot \frac{M^{\frac{5}{2}}}{(\log M)^2}$$

for large enough M. ■

Lemma 16 *Let M and K be positive constants. For a fixed positive integer a, let $\mathcal{T}_{a,M,K}$ denote the set of pairs of primes (x, y) such that $\frac{M}{2} \le x \le M$, $|x^2 - a \cdot y| \le 5x^{3/2}$ and $y | x^k - 1$ for some $k \le K$. Then $|\mathcal{T}_{a,M,K}| < \frac{45}{8} M^{3/2}(K+1)^2 \log(5^{3/2}M)$.*

Proof For every nonzero integer h with $|h| \le 5M^{3/2}$, let $\mathcal{B}_h^{(e)}$ be the set of primes y such that $y|h^{k/2} - 1$ for some even integer k with $0 < k \le K$. Since $h^{k/2} - 1$ has fewer than $\log(|h|^{k/2})$ distinct prime divisors, we have

$$\begin{aligned}
|\mathcal{B}_h^{(e)}| &< \sum_{\substack{k=2 \\ k \text{ even}}}^{K} \frac{k}{2} \log |h| \le \frac{1}{2}\left(\frac{K}{2}\right)\left(\frac{K}{2} + 1\right) \log |h| \\
&\le \frac{1}{2}\left(\frac{K}{2}\right)\left(\frac{K}{2} + 1\right) (3/2) \log(5^{3/2}M) \\
&\le \frac{3}{16} K(K+2) \log(5^{3/2}M).
\end{aligned}$$

Now for the same h, let $\mathcal{B}_h^{(o)}$ denote the set of primes y such that $y|h^k - 1$ for some odd integer k with $0 < k \le K$. Since $h^k - 1$ has fewer than $\log(|h|^k)$ distinct prime divisors,

$$\begin{aligned}
|\mathcal{B}_h^{(o)}| &< \sum_{\substack{k=1 \\ k \text{ odd}}}^{K} k \log |h| \le \frac{\lceil \frac{K}{2} \rceil (K+1)}{2} \log |h| \\
&\le \frac{1}{4}(K+1)^2 \, (3/2) \log\left(5^{3/2}M\right) \\
&= \frac{3}{8}(K+1)^2 \log\left(5^{3/2}M\right).
\end{aligned}$$

Let $\mathcal{B}_h$ be the set of pairs of primes (x, y) such that $x^2 - a \cdot y = h$. When k is even, we have

$$h^{k/2} = \left(x^2 - a \cdot y\right)^{k/2} = x^k + y \cdot (\text{polynomial in } x \text{ and } y);$$

thus $y | h^{k/2} - 1$ is equivalent to $y | x^k - 1$. Similarly, when k is odd, $y | x^k - 1$ implies $y | x^{2k} - 1$, which again implies $y | h^k - 1$. Therefore, we must have

$$\begin{aligned} |\mathcal{B}_h| &\leq |\mathcal{B}_h^{(e)}| + |\mathcal{B}_h^{(o)}| \\ &\leq \frac{3}{16} K(K+2) \log\left(5^{3/2} M\right) + \frac{3}{8}(K+1)^2 \log\left(5^{3/2} M\right) \\ &< \frac{9}{16}(K+1)^2 \log(5^{3/2} M). \end{aligned}$$

Summing over all such integer h and note that $\frac{M}{2} \leq x \leq M$, we have

$$|\mathcal{T}_{a,M,K}| \leq \sum_{0 < |h| \leq 5M^{3/2}} |\mathcal{B}_h| < \frac{45}{8} M^{3/2} (K+1)^2 \log(5^{3/2} M).$$

■

Lemma 17 *Let* $\widetilde{\mathcal{S}}_{H,c,M}$ *denote the set of pairs of primes* (x, y) *such that* $\frac{M}{2} \leq x \leq M$ *and* $|x^2 - a \cdot y| \leq c \cdot x^{3/2}$ *for some* $a \in \mathbb{Z}$*,* $1 \leq a \leq H$*. Let* $\widetilde{\mathcal{T}}_{H,M,K}$ *denote the set of pairs of primes* (x, y) *such that* $\frac{M}{2} \leq x \leq M$*,* $|x^2 - a \cdot y| \leq 5x^{3/2}$ *for some* $a \in \mathbb{Z}$*,* $1 \leq a \leq H$*, and* $y | x^k - 1$ *for some* $k \leq K$*. If the R.H. holds, then for large* M*,*

$$\frac{\widetilde{\mathcal{T}}_{H,M,K}}{\widetilde{\mathcal{S}}_{H,c,M}} < c' \frac{H \cdot (K+1)^2 (\log M)^3}{c \cdot M}$$

for an effectively computable positive constant c'*. A possible choice of such a constant is* $c' = 90$ *.*

Proof Let a be an integer such that $1 \leq a \leq H$. By Lemma 15 and Lemma 16, we have

$$\begin{aligned} \frac{\mathcal{T}_{a,M,K}}{\mathcal{S}_{a,c,M}} &< \frac{\frac{45}{8} M^{3/2} (K+1)^2 \log(5^{3/2} M)}{\frac{1}{15} \cdot \frac{c}{a} \cdot \frac{M^{\frac{5}{2}}}{(\log M)^2}} \\ &< 90 \cdot \frac{a \cdot (K+1)^2 (\log M)^3}{c \cdot M} \\ &< 90 \cdot \frac{H \cdot (K+1)^2 (\log M)^3}{c \cdot M} \end{aligned}$$

for M large enough. Note that $\widetilde{\mathcal{T}}_{H,M,K} = \sum_{1 \leq a \leq H} \mathcal{T}_{a,M,K}$ and $\widetilde{\mathcal{S}}_{H,c,M} = \sum_{1 \leq a \leq H} \mathcal{S}_{a,c,M}$. Hence we have

$$\frac{\widetilde{\mathcal{T}}_{H,M,K}}{\widetilde{\mathcal{S}}_{H,c,M}} < 90 \cdot \frac{H \cdot (K+1)^2 (\log M)^3}{c \cdot M}$$

for large M.

$$\blacksquare$$

According to a result of D. Jao et al. [80], the discrete logarithm problem has the same difficulty for all elliptic curves over a given finite field with the same order. With this, it is reasonable to conjecture that the same result holds for genus 2 curves, i.e., the discrete logarithm problem has the same difficulty for all genus 2 hyperelliptic curves C over given a finite field $\mathbb{F}_{\|}$ such that $\#Jac_{\mathbb{F}_q}(C)$ is the same. From this cryptographic point of view, in the following theorem, we treat all genus 2 curves C over a given prime field $\mathbb{F}_p$ as the same curve, if all $Jac_{\mathbb{F}_q}(C)$ have the same cardinality.

Theorem 18 *Assume the Riemann Hypothesis. Let H and K be positive constants. Let (p, C) be a randomly (w.r.t. uniform distribution) chosen pair in which p is a prime in the interval $[\frac{M}{2}, M]$ and C is a genus 2 hyperelliptic curve defined over $\mathbb{F}_p$ such that $\#Jac_{\mathbb{F}_p}(C) = h \cdot r$ with $1 \leq h \leq H$ and r prime. For M large enough, the probability that C is pairing-friendly with respect to parameters H and K is less than*

$$c'' \frac{H \cdot (K+1)^2 (\log M)^3}{M}$$

for an effectively computable positive constant c''.

Proof The Riemann Hypothesis for abelian varieties over finite fields, proved by A. Weil in [42], implies the Hasse-Weil bound for genus 2 curves, i.e.,

$$\#Jac_{\mathbb{F}_p}(C) \in \left[(\sqrt{p}-1)^4, (\sqrt{p}+1)^4\right].$$

For p large enough, we have $\#Jac_{\mathbb{F}_p}(C) \in \left[p^2 - 5p^{3/2}, p^2 + 5p^{3/2}\right]$.

Let $c = \frac{1}{9}$. By Proposition 2.4 of [81], almost all integers $z \in \left[p^2 - cp^{3/2}, p^2 + cp^{3/2}\right]$ can be assumed as the cardinality of the Jacobian of a genus 2 hyperelliptic curve (given by a quintic or sextic polynomial) over $\mathbb{F}_p$.

The conclusion then follows from Lemma 17. Note that we can choose $c'' = 10c'$, where c' is the constant from Lemma 17.

■

Theorem 18 says there are very few pairing-friendly genus 2 hyperelliptic curves with respect to parameters H and K much smaller than p.

5.4 Algorithms for Generating Pairing-friendly Genus 2 Curves over Prime Fields

Let k be a desired embedding degree. Let C be a genus 2 hyperelliptic curve defined over a finite field $\mathbb{F}_p$ whose Jacobian over $\mathbb{F}_p$ has a subgroup of order r such that $Jac_{\mathbb{F}_p}(C)$ has embedding degree k with respect to r. The ratio of the bit length of $\#Jac_{\mathbb{F}_p}(C)$ to the bit length of r is a good measure of efficiency in pairing-based cryptography. If we define

$$\rho = 2\log(p)/\log(r),$$

then this value is a good approximation of the above ratio. In many cryptographic applications, we prefer this value to be close to 1.

In [71], a method to generate genus 2 curves with ordinary Jacobians over prime fields with low embedding degrees is proposed. An important part of this method is a parameterization of the CM field. The method generates curves with value $\rho \approx 8$. We propose another way of generating good parameters, without parameterizing the CM field, which gives a similar ρ value.

We continue to use the same notation and assumptions as in Chapter 4. Again we let $K := \mathbb{Q}(\eta)$ be the fixed quartic CM field and want to construct a genus 2 hyperelliptic curve C over a prime field $\mathbb{F}_p$ such that $Jac_{\mathbb{F}_p}(C)$ has CM by K, and such that $Jac_{\mathbb{F}_p}(C)$ has a subgroup of prime order r and $Jac_{\mathbb{F}_p}(C)$ has a prescribed embedding degree k with respect to r. For cryptographic applications, we also need p and r to be large. We will present the algorithm for the case $d \equiv 2, 3 \pmod 4$ in this thesis. The case $d \equiv 1 \pmod 4$ can be treated similarly.

In the case $d \equiv 2, 3 \pmod 4$, such a curve can be constructed if we can find a simultaneous integral solution $(c_1, c_2, c_3, c_4, p, r)$, in which p and r are large prime numbers, to the following system of equations:

$$\begin{aligned} c_1^2 + c_2^2 d + c_3^2 a + c_4^2 ad + 2c_3c_4bd &= p && (5.1) \\ 2c_1c_2 + 2c_3c_4a + c_3^2 b + c_4^2 bd &= 0 && (5.2) \\ (p+1)^2 - 4c_1(p+1) + 4(c_1^2 - dc_2^2) &\equiv 0 \pmod r && (5.3) \\ \Phi_k(p) &\equiv 0 \pmod r. && (5.4) \end{aligned}$$

Here a, b, d and k are fixed, and $\Phi_k(x)$ is the k^{th} cyclotomic polynomial. Equations (5.1) and (5.2) mean that the prime p corresponds to a good Weil number, as in Chapter 4. Equation (5.3) makes sure that the Jacobian has a subgroup of prime order r. Equation (5.4) guarantees that the Jacobian of the curve has an embedding degree with respect to r at most k.

Algorithm 7 Generating pairing parameters for $K = \mathbb{Q}(\eta)$, $d \equiv 2, 3 \pmod 4$

```
Input: Integers $a, b, d$ with $d > 0$ squarefree, $d \equiv 2, 3 \pmod 4$, $a^2 - b^2 d > 0$ not
    a square; a prescribed embedding degree $k$; a bit size $n$ of the desired subgroup
    order; maximum numbers of trials, $M_1$ and $M_2$.
Output: Integers $c_1, c_2, c_3, c_4$, prime numbers $p$ and $r$, where $r$ has $n$ bits, satisfying
    Equations (5.1), (5.2), (5.3), (5.4); or "Not found."
 1: Let $c_1 = \pm 1$.
 2: repeat
 3:   Choose a prime number $r$ of $n$ bits such that $r \equiv 1 \pmod k$.
 4:   With $c_1$ fixed as above, try to solve the system of equations given by (5.1), (5.2),
      (5.3), (5.4) over the finite field $\mathbb{F}_r$ for a simultaneous solution $(\bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{p})$.
 5:   if Such a solution exists then
 6:     repeat
 7:       Choose lifts $c_3$ and $c_4$ of $\bar{c}_3$ and $\bar{c}_4$ to $\mathbb{Z}$ such that $f := bc_3^2 + 2ac_3c_4 + bdc_4^2$
          is even. Set $c_2 = -c_1 f/2$.
 8:       Let $p = ac_3^2 + 2bdc_3c_4 + 2adc_4^2 + 1 + dc_2^2$.
 9:       if $p$ is prime then
10:         Return $(c_1, c_2, c_3, c_4, p, r)$.
11:       end if
12:     until Lines 7 through 11 have been tried $M_2$ times.
13:   end if
14: until $M_1$ primes $r$ have been tried.
15: Return "Not found."
```

Theorem 19 *If* $(c_1, c_2, c_3, c_4, p, r)$ *is returned by Algorithm 7, then it provides a solution to the system of equations (5.1), (5.2), (5.3), (5.4).*

Proof It is clear that if $(c_1, c_2, c_3, c_4, p, r)$ is returned, then Equations (5.3) and (5.4) are automatically satisfied. Equations (5.1) and (5.2) are satisfied by the constructions in Step 7 and 8. Step 9 ensures that p is prime. ■

Depending on p and $\mathcal{O}_K$, there are 2 or 4 possibilities for the group order $\#Jac_{\mathbb{F}_q}(C)$ [57,58]. However, we are only interested in the curve C whose Jacobian has an exact group order given by

$$N = (p+1)^2 - 4c_1(p+1) + 4(c_1^2 - dc_2^2).$$

Algorithm 7 is difficult to analyze because we do not know how likely a solution is found in Step 4. However, experimental result shows that the algorithm returns valid parameters with high probability.

Example 3. In the case of $a = 2, b = -1, d = 2$, some suitable pairing parameters are found as in Appendix B.1, where r are 160, 256, 512 and 1024 bits, respectively. The computations were performed by the computer algebra system MAGMA [53]. Note that this is the case that $K = \mathbb{Q}[i\sqrt{2-\sqrt{2}}] \neq \mathbb{Q}(\xi_5)$ is Galois, so there are only two possibilities for the group order $\#Jac_{\mathbb{F}_p}(C)$ [58], namely,

$$N_1 = (p+1)^2 - 4c_1(p+1) + 4(c_1^2 - dc_2^2),$$

which corresponds to the curve we need, or

$$N_2 = 2(p+1)^2 + 8(c_1^2 - c_2^2 d) - N_1,$$

which corresponds to a quadratic twist of the desired curve.

5.5 Generating Pairing Parameters with Polynomial Parameterization

The parameter c_1 produced by Algorithm 7 is always ±1 and the size of c_2 dominates that of c_1, c_3 and c_4. In fact, this is not necessary. We can modify the search method with the idea of polynomial parameterization and produce pairing parameters with c_1, c_2, c_3 and c_4 are roughly of the same size. The algorithm is given as follows.

Algorithm 8 Generating pairing parameters for $K = \mathbb{Q}(\eta)$, $d \equiv 2, 3 \pmod 4$ with polynomial parameterization

Input: Integers a, b, d with $d > 0$ squarefree, $d \equiv 2, 3 \pmod 4$, $a^2 - b^2 d > 0$ not a square; a prescribed embedding degree k; a bit size n of the desired subgroup order; maximum numbers of trials, M_1 and M_2.

Output: Integers c_1, c_2, c_3, c_4, prime numbers p and r, where r has n bits, satisfying Equations (5.1), (5.2), (5.3), (5.4); or "Not found."

```
1:  Choose degree 2 bivariate polynomials $C_3(x, y)$ and $C_4(x, y) \in \mathbb{Z}[x, y]$ such that
    there is a factorization in $\mathbb{Z}[x, y]$
        $bC_3^2 + 2aC_3C_4 + bdC_4^2 = U \cdot V,$
    where $U$ and $V$ are bivariate polynomials of degree 2. Let $C_1(x, y) = U(x, y)$ and
    $C_2(x, y) = -\frac{1}{2}V(x, y)$.
2:  repeat
3:      Choose a prime number $r$ of $n$ bits such that $r \equiv 1 \pmod k$.
4:      Try to solve the system of equations given by (5.2), (5.3), (5.4), with $c_i$ replaced
        by $C_i(x, y), i = 1, 2, 3, 4$, over the finite field $\mathbb{F}_r$ for a simultaneous solution
        $(\bar{x}, \bar{y}, \bar{p})$.
5:      if Such a solution exists then
6:          repeat
7:              Choose lifts $x$ and $y$ of $\bar{x}$ and $\bar{y}$ to $\mathbb{Z}$ such that $c_i := C_i(x, y), i = 1, 2, 3, 4$
                are all integers. Let $p = ac_3^2 + 2bdc_3c_4 + 2adc_4^2 + c_1^2 + dc_2^2$.
8:              if $p$ is prime then
9:                  Return $(c_1, c_2, c_3, c_4, p, r)$.
10:             end if
11:         until Lines 7 through 10 have been tried $M_2$ times.
12:     end if
13: until $M_1$ primes $r$ have been tried.
14: Return "Not found."
```

Similarly, we have

Theorem 20 *If* $(c_1, c_2, c_3, c_4, p, r)$ *is returned by Algorithm 8, then it provides a solution to the system of equations (5.1), (5.2), (5.3), (5.4).*

In Algorithm 8, it is clear that we need $\gcd(C_1, C_2, C_3, C_4) = 1 \in \mathbb{Z}[x, y]$ so that a prime p can be found.

Example 3. Let $C_3(x, y) = C_4(x, y) = xy$, $C_1(x, y) = x^2$ and $C_2(x, y) = -(a + b(1 + d)/2)y^2$. Then they satisfy $bC_3^2 + 2aC_3C_4 + bdC_4^2 + 2C_1C_2 = 0$. Using these polynomials in the above algorithm, we have found for $K = \mathbb{Q}(i\sqrt{2-\sqrt{2}})$ (i.e., $a = 2, b = -1, d = 2$) parameters in which r are 160, 256, 512 and 1024 bits, respectively. These parameters are presented in Appendix B.2.

Since x and y are roughly of the size of r, the value of p obtained by this method is $\approx r^4$. It is thus a natural thought that if we parameterize the polynomials $C_i(x, y)$ with degree 1 polynomials in $\mathbb{Z}[x, y]$, then the size of p may be reduced to $\approx r^2$. The following proposition shows that such parameterizations will not succeed.

Proposition 21 *Let* a, b, d *be integers such that* d *is squarefree and* $a^2 - b^2d > 0$ *is not a square. Let* $f(X, Y) = bX^2 + 2aXY + bdY^2$ *be a bivariate polynomial in* $\mathbb{Q}[X, Y]$*. Let* F, G *be polynomials of total degree* 1 *in* $\mathbb{Q}[X_1, X_2, \ldots, X_n]$ *such that* F *and* G *are not associated with one another. Then* $f(F, G)$ *is irreducible in* $\mathbb{Q}[X_1, X_2, \ldots, X_n]$*.*

Proof First we note that $b \neq 0$, as indicated by the condition that $a^2 - b^2d > 0$ is not a square. Let $D = a^2 - b^2d$. Let $\alpha = -a/b + \sqrt{D}/b$ and $\beta = -a/b - \sqrt{D}/b$. Then $f(X, Y)$ can be factored over $\bar{\mathbb{Q}}$ as

$$f(X, Y) = bX^2 + 2aXY + bdY^2 = b(X - \alpha Y)(X - \beta Y),$$

where $\bar{\mathbb{Q}}$ is an algebraic closure of $\mathbb{Q}$.

Let F and G be polynomials of total degree 1 in $\mathbb{Q}[X_1, X_2, \ldots, X_n]$. Write

$$F(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} f_i X_i + f_0,$$
$$G(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} g_i X_i + g_0,$$

where $f_i, g_i \in \mathbb{Q}$. Suppose $f(F, G)$ is reducible in $\mathbb{Q}[X_1, X_2, \ldots, X_n]$. Then we can write

$$f(F, G) = bH_1 \cdot H_2,$$

where $H_j = \sum_{i=1}^{n} h_i^{(j)} X_i + h_0^{(j)} \in \mathbb{Q}[X_1, X_2, \ldots, X_n]$, $j = 1, 2$, both of total degree 1.

Now we have

$$b(F - \alpha G)(F - \beta G) = f(F, G) = bH_1 \cdot H_2.$$

Note that $\mathbb{Q}(\sqrt{D})[X_1, X_2, \ldots, X_n]$ is a uniform factorization domain. Because $F - \alpha G$, $F - \beta G$, H_1 and H_2 are of degree 1, they are irreducible. without of loss of generality, we may assume

$$F - \alpha G = \gamma H_1, \tag{5.5}$$

for some $\gamma \in \mathbb{Q}(\sqrt{D})^{\times}$. We can write $\gamma = s + t\sqrt{D}$ with $s, t \in \mathbb{Q}$ and $t \neq 0$. Here we require $t \neq 0$ as the polynomial on the left hand side of Equation (5.5) is in $\mathbb{Q}(\sqrt{D})[X_1, X_2, \ldots, X_n] \backslash \mathbb{Q}[X_1, X_2, \ldots, X_n]$.

Equation (5.5) gives

$$F - (-a/b + \sqrt{D}/b)G = (s + t\sqrt{D})H_1. \tag{5.6}$$

Equating the coefficients of X_i and the constant terms on both sides of the above equation, we obtain

$$f_i + (a/b)g_i + (g_i/b)\sqrt{D} = s \cdot h_i^{(1)} + t \cdot h_i^{(1)}\sqrt{D}, \quad 0 \leq i \leq n. \tag{5.7}$$

This in turn gives

$$f_i + (a/b)g_i = s \cdot h_i^{(1)}, \tag{5.8}$$

$$g_i/b = t \cdot h_i^{(1)}. \tag{5.9}$$

If $g_i = 0$ for some i, we must have $h_i^{(1)} = 0$ by (5.9), which again implies $f_i = 0$ by (5.8). Otherwise, if $g_i \neq 0$, we can divide both sides of (5.8) and (5.9) to obtain

$$b(f_i/g_i) = s/t,$$

thus

$$f_i/g_i = s/(b \cdot t). \tag{5.10}$$

Therefore, for all $0 \leq i \leq n$, we have $f_i = c \cdot g_i$, where the constant $c = s/(b \cdot t) \in \mathbb{Q}$. Hence $F = c \cdot G$, i.e., F and G are associated. ■

An alternative way of polynomial parameterization in Step 1 of Algorithm 8 is to use degree 1 and degree 2 polynomials for $C_3(x, y)$ and $C_4(x, y)$. This will produce different kinds of c_i's, but the resulting ρ value is still approximately 8 in general. The on-going research is aiming at reducing further the value of ρ. Our next goal is to find an efficient algorithm that produces a ρ value close to 4 for the CM method. Our ultimate goal is to find an efficient method that gives genus 2 curves over large prime fields with $\rho \approx 1$.

5.6 Updates on Related Research and Future Work

In 2002, K. Rubin and A. Silverberg [82] showed that supersingular Jacobians of genus 2 hyperelliptic curves have small embedding degrees (≤ 12). In 2007, L. Hitt [83] presented, for characteristic 2, the existence of families of genus 2 curves with small embedding degree and small ρ value (< 2). D. Freeman [71] gave a method in 2007 for constructing genus 2 curves with ordinary Jacobians over prime fields, with $\rho \approx 8$. Freeman's method uses parameterization of the CM fields to obtain conditions that lead to the result.

We describe in this chapter a new method for generating pairing-friendly parameters, without parameterizing the CM fields. The method is easy to understand and easy to implement.

After the research described in this chapter was done, other methods [84, 85] for finding pairing-friendly parameters were proposed, and they produce parameters with smaller ρ values (≈ 4 or ≤ 4).

The next step of our research is to allow non-integral values for the coefficients of the Frobenius element, and try to find further relations between the parameters. Based on this, we want to find better solutions to a system of equations as described in this chapter, which provide smaller ρ values.

6. CONCLUSIONS AND FUTURE WORK

The thesis contributes to both applied and theoretical cryptography. We summarize the contributions as follows.

In Chapter 2, a time-bound hierarchical key management scheme for access control is proposed. Deployment of elliptic curve cryptography makes the scheme resistant against attacks that break prior proposals of such schemes. The scheme is designed for broadcasting of encrypted data, and is useful in real-world applications like electronic newspaper subscription and Pay TV.

In Chapter 3, explicit doubling formulas are presented for genus 2 hyperelliptic curves in the real model. The most general case and special cases of divisor doubling are handled by these formulas. They are useful for efficient implementation of cryptographic protocols using genus 2 real hyperelliptic curve. Equivalent transformations for obtaining short equations of genus 2 real hyperelliptic curves are also investigated and presented. This extends the existing work on equivalent transformations for imaginary hyperelliptic curves.

Chapter 4 shows a method which generates suitable parameters for the complex multiplication construction of genus 2 curves that can be used in cryptography. The proposed method uses polynomial parameterization to improve the efficiency over earlier published literature, by avoiding factorization of large integers. Analysis of the method is presented based on the Bateman-Horn heuristics. We also give a new conjecture that extends the existing work. The new conjecture deals with the case of two polynomials and the “almost prime” condition. Examples are given to provide numerical evidence of our conjecture.

In Chapter 5, following a quantitative analysis of the scarcity of pairing-friendly genus 2 curves, a method of finding parameters for generating such curves via the genus 2 complex multiplication construction is presented. The method finds parameters that give $\rho \approx 8$. The analysis about the scarcity of pairing-friendly genus 2 curve improves a prior heuristic result. And our algorithm for finding parameters is easy to understand and to implement.

We describe some of the directions for further research as follows.

The work in Chapter 2 can be extended to achieve a complete security proof. It is also desirable to construct an efficient time-bound hierarchical key management scheme that does not use a tamper-resistant device. Implementation and performance evaluation of the scheme are also to be done in future research.

Future work in Chapter 3 includes optimization of current formulas, derivation of more efficient explicit addition, doubling, and baby step formulas for genus 2 real hyperelliptic curve using coordinates other than affine coordinates. Efficient implementation and application oriented performance evaluation is another direction of further research.

Research work in Chapters 4 and 5 can be regarded as being in the same framework as finding cryptographically strong parameters in the construction of genus 2 curves. Further research needs to improve the current methods, and/or discover new methods so that parameters can be produced more efficiently, and that pairing-friendly parameters are generated with smaller ρ value. Again, implementation is part of the future work.

LIST OF REFERENCES

LIST OF REFERENCES

[1] A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[2] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[3] `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`. PKCS #1: RSA Cryptography Standard.

[4] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Information Theory*, 24:106–110, 1978.

[5] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[6] V. S. Miller. Use of elliptic curves in cryptography. In *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[7] A. M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Theory and Application of Cryptographic Techniques*, pages 224–314, 1984.

[8] L. M. Adleman, J. DeMarrais, and M-D A. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In *ANTS-I: Proceedings of the First International Symposium on Algorithmic Number Theory*, pages 28–40, London, UK, 1994. Springer-Verlag.

[9] V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Mathematics of Computation*, 68(226):807–822, 1999.

[10] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Advances in Cryptology – Eurocrypt2000*, volume 1807, pages 19–34, Berlin, 2000. Springer-Verlag.

[11] A. Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Mathematics of Computation*, 71(238):729–742, 2002.

[12] N. Theriault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 75–92. Springer, 2003.

[13] P. Gaudry, E. Thome, N. Theriault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, 76:475–492, 2007.

[14] H. Cohen and G. Frey. *Handbook of Elliptic and Hyperelliptic Curve Cryptography: Theory and Practice.* CRC Press, 2005.

[15] S. Galbraith, S. Paulus, and N. Smart. Arithmetic on superelliptic curves. *Mathematics of Computation*, 71(237):393–405, 2002.

[16] E. Bertino, N. Shang, and S. S. Wagstaff, Jr. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Transactions on Dependable and Secure Computing*, 5(2):65–70, 2008.

[17] R. Scheidler, A. Stein, and H.C. Williams. Key-exchange in real quadratic congruence function fields. *Design, Codes and Cryptography*, 7:153–174, 1996.

[18] W.G. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):182–188, Jan./Feb. 2002.

[19] X. Yi and Y. Ye. Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1054–1055, Jul./Aug 2003.

[20] H.Y. Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1302–1304, October 2004.

[21] X. Yi. Security of Chien's efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1298–1299, September 2005.

[22] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. *5th International Workshop on Security Protocols (LNCS 1361)*, pages 125–136, 1997.

[23] E. Bertino, B. Carminati, and E. Ferrari. A temporal key management scheme for secure broadcasting of XML documents. *CCS'02*, pages 31–40, November 2002.

[24] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems*, 23(3):231–285, September 1998.

[25] L. C. Washington. *Elliptic Curves, Number Theory and Cryptography.* Chapman & Hall/CRC, 2003.

[26] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC). `http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf`.

[27] Advanced Encryption Standard. `http://csrc.nist.gov/CryptoToolkit/aes/`.

[28] Trusted Platform Module. https://www.trustedcomputinggroup.org/groups/tpm/.

[29] Evan R. Sparks. A Security Assessment of Trusted Platform Modules. Technical Report TR2007-597, Dartmouth College, Computer Science, Hanover, NH, June 2007.

[30] A. Jurisic and A. J. Menezes. Elliptic curves and cryptography. *Dr. Dobb's Journal*, pages 23–36, April 1997.

[31] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[32] `http://www.cardwerk.com/smartcards/smartcard_technology.aspx`. Web article: Smart Card Technology.

[33] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, 1989.

[34] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication, and Computing*, 15:295–328, 2005.

[35] S. Erickson, M. J. Jacobson, Jr., N. Shang, S. Shen, and A. Stein. Explicit formulas for real hyperelliptic curves of genus 2 in affine representation. In *WAIFI*, pages 202–218, 2007.

[36] M. Rosen. *Number Theory in Function Fields.* Springer-Verlag, New York, NY, USA, 2002.

[37] P. Lockhart. On the discriminant of a hyperelliptic curve. *Transactions of the American Mathematical Society*, 342(2):729–752, April 1994.

[38] A. Enge. How to distinguish hyperelliptic curves in even characteristic. In *Public-Key Cryptography and Computational Number Theory*, pages 49–58, 2001.

[39] T. Lange. *Efficient Arithmetic on Hyperelliptic Curves.* PhD thesis, Universität-Gesamthochschule Essen, 2002.

[40] D. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of Computation*, 48(177):95–101, January 1987.

[41] J. H. Silverman. *The Arithmetic of Elliptic Curves.* Springer-Verlag, 1986.

[42] A. Weil. Variétés Abéliennes et Courbes Algébriques. *Paris, Hermann*, 1948.

[43] D. Bernstein and T. Lange. Elliptic vs. hyperelliptic, parts 1, 2. Talks given at The 10th Workshop on Elliptic Curve Cryptography (ECC 2006).

[44] N. Shang. G2HEC: A Genus 2 Crypto C++ Library. http://www.math.purdue.edu/˜nshang/libg2hec.html.

[45] V. Shoup. NTL: A Library for doing Number Theory. http://www.shoup.net/ntl/.

[46] S. Paulus and H.-G. Rück. Real and imaginary quadratic representations of hyperelliptic function fields. *Mathematics of Computation*, 68:1233–1241, 1999.

[47] M. J. Jacobson, Jr., R. Scheidler, and A. Stein. Fast arithmetic on hyperelliptic curves via continued fraction expansions. In *Advances in Coding Theory and Cryptology, Series on Coding Theory and Cryptology 3. Ed*, pages 201–244, 2007.

[48] A. Stein. Sharp upper bounds for arithmetics in hyperelliptic function fields. *Journal of the Ramanujan Mathematical Society*, 9-16(2):1–86, 2001.

[49] M. J. Jacobson, Jr., R. Scheidler, and A. Stein. Cryptographic protocols on real hyperelliptic curves. *Advances in Mathematics of Communications*, 1(2):197–221, 2007.

[50] S. Erickson, M. J. Jacobson, Jr., N. Shang, S. Shen, and A. Stein. More explicit formulas for real hyperelliptic curves of genus 2. In preparation.

[51] S. Shen. *Finite Fields of Low Characteristic in Elliptic Curve Cryptography*. PhD thesis, Purdue University, 2007.

[52] R. Harley. Efficient algorithm for computing the group law in the jacobian of a genus-2 curve. http://cristal.inria.fr/ harley/hyper/adding.text.

[53] W. Bosma, J. Cannon, and C. Playoust. The MAGMA algebra system I: the user language. *J. Symb. Comput.*, 24(3-4):235–265, 1997.

[54] N. Koblitz, A. J. Menezes, Y-H Wu, and R. J. Zuccherato. *Algebraic aspects of cryptography*. Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[55] P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. In *ANTS*, pages 313–332, 2000.

[56] E. Furukawa, M. Kawazoe, and T. Takahashi. Counting points on the Jacobian variety of a hyperelliptic curve defined by $y^2 = x^5 + ax$ over a prime field, 2002.

[57] K. Eisenträger and K. Lauter. A CRT algorithm for constructing genus 2 curves over finite fields, 2004. `http://arxiv.org/abs/math.NT/0405305v2`.

[58] A. Weng. Constructing hyperelliptic curves of genus 2 suitable for cryptography. *Mathematics of Computation*, 72(241):435–458, 2002.

[59] P. Van Wamelen. Examples of genus two CM curves defined over the rationals. *Mathematics of Computation*, 68(225):307–320, 1999.

[60] P. Gaudry, T. Houtmann, D. Kohel, C. Ritzenthaler, and A. Weng. The 2-adic CM method for genus 2 curves with application to cryptography. In *ASIACRYPT*, pages 114–129, 2006.

[61] J-F Mestre. Construction de courbes de genre 2 à partir de leurs modules. (Construction of genus 2 curves starting from their moduli). Effective methods in algebraic geometry, Proc. Symp., Castiglioncello/Italy 1990, Prog. Math. 94, 313-334 (1991)., 1991.

[62] G. Cardona and J. Quer. Field of moduli and field of definition for curves of gens 2. In *Computational aspects of algebraic curves*, volume 13, pages 71–83, 2005.

[63] W. Gautschi. *Numerical Analysis: An Introduction*. Birkhauser Boston Inc., Cambridge, MA, USA, 1997.

[64] L. M. Adleman and A. M. Odlyzko. Irreducibility testing and factorization of polynomials. *Mathematics of Computation*, 41(164):699–709, 1983.

[65] The PARI Group, Bordeaux. *PARI/GP, version* `2.1.7`, 2005. Available from `http://pari.math.u-bordeaux.fr/`.

[66] P. T. Bateman and R. A. Horn. A heuristic asymptotic formula concerning the distribution of prime numbers. *Mathematics of Computation*, 16(79):363–367, July 1962.

[67] P. T. Bateman and R. A. Horn. Primes represented by irreducible polynomials in one variable. *Proc. Sympos. Pure Math.*, 8:119–132, 1965.

[68] A. Schinzel and W. Sierpinski. Sur certaines hypotheses concernant les nombres premiers. *Actu Arith.*, 4:185–218, 1958. erratum, v. 5, 1959, p. 259.

[69] A. Schinzel. Remarks on the paper "sur certaines hypotheses concernant les nombres premiers. *Acta Arith.*, 7:1–8, 1961/1962.

[70] S. D. Galbraith, J. Mckee, and P. Valena. Ordinary abelian varieties having small embedding degree. In *Proc. Workshop on Mathematical Problems and Techniques in Cryptology*, pages 29–45, 2004.

[71] D. Freeman. Constructing pairing-friendly genus 2 curves over prime fields with ordinary Jacobians. In *Proceedings of Pairing-Based Cryptography (Pairing 2007)*, volume 4575 of *LNCS*, pages 152–176. Springer, 2007.

[72] D. Boneh and C. Gentry. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of Eurocrypt 2003, volume 2656 of LNCS*, pages 416–432. Springer-Verlag, 2003.

[73] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[74] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *J. Cryptol.*, 17(4):297–319, 2004.

[75] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[76] R. Dupont and A. Enge. Provably secure non-interactive key distribution based on pairings. *Discrete Appl. Math.*, 154(2):270–276, 2006.

[77] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes - Okamoto - Vanstone algorithm. *Journal of Cryptology*, 11(2):141–145, 1998.

[78] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms.* The MIT Press, August 1996.

[79] L. Schoenfeld. Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. II. *Mathematics of Computation*, 30(134):337–360, April 1976.

[80] D. Jao, S. D. Miller, and R. Venkatesan. Do all elliptic curves of the same order have the same difficulty of discrete log? In *ASIACRYPT*, pages 21–40, 2005.

[81] H. W. Lenstra, Jr, J. Pila, and C. Pomerance. A hyperelliptic smoothness test, II. *Proc. London Math. Soc.*, 84(1):105–146, 2002.

[82] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 336–353, London, UK, 2002. Springer-Verlag.

[83] L. Hitt. Families of genus 2 curves with small embedding degree. Cryptology ePrint Archive, Report 2007/001, 2007.

[84] D. Freeman. A Generalized Brezing-Weng Algorithm for Constructing Pairing-Friendly Ordinary Abelian Varieties. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 146–163. Springer, 2008.

[85] M. Kawazoe and T. Takahashi. Pairing-friendly hyperelliptic curves of type $y^2 = x^5 + ax$. In *Symposium on Cryptography and Information Security (SCIS)*, 2008.

APPENDICES

A. Parameters for Discrete Log Based Cryptography

In the following, we present some parameters found by our method. The primes $p(x_0)$, corresponding to value x_0, are of 128 bits and the group orders $N_1(x_0)$ or $N_2(x_0)$ are almost prime (in this case, a product of a positive integer < 2000 and a prime number).

$\underline{a = 20, b = 1, d = 19.}$

$$
\begin{aligned}
c_1(x) &= -(5/2)x^2 - 24x - 39/2,\\
c_2(x) &= 12x^2 + 42x + 5,\\
c_3(x) &= x^2 + 2x + 4,\\
c_4(x) &= x^2 + 9x + 1,\\
p(x) &= (12721/4)x^4 + 26610x^3 + (138247/2)x^2 + 17520x + 6829/4,\\
N_1(x) &= (161823841/16)x^8 + 169252905x^7 + (4591135697/4)x^6 + 3790760034x^5\\
&\quad + (45798567295/8)x^4 + 2521811013x^3 + (2200243933/4)x^2 + 61359456x + 48815721/16,\\
N_2(x) &= (161823841/16)x^8 + 169252905x^7 + (4590881277/4)x^6 + 3789617226x^5\\
&\quad + (45742665623/8)x^4 + 2504037741x^3 + (2143518849/4)x^2 + 58298352x + 44551929/16.\\
x_0 &= 548050991,\\
p(x_0) &= 286909637977764067855221276777587727961,\\
N_2(x_0) &= 82317140364531637515130621054159952023115110860990352140958946602968889685624\\
&= 2^3 \cdot 103 \cdot 99899442189965579508653666327864019445528047161396058423493867236612\\
&\quad 730201.\\
x_0 &= 507822535,\\
p(x_0) &= 211499402528761325611378043169347082601,\\
N_1(x_0) &= 44731997270023012615201443067044482876392989995456465952479290805233356211416\\
&= 2^3 \cdot 3^2 \cdot 62127773986143073076668670926450670661656930549245091600665681673935\\
&\quad 2169603.
\end{aligned}
$$

$\underline{a = 49, b = 2, d = 43.}$

$$\begin{aligned}
c_1(x) &= -19x^2 - 81x - 57, \\
c_2(x) &= 25x^2 + 77x + 37, \\
c_3(x) &= 3x^2 + 9x + 7, \\
c_4(x) &= 2x^2 + 8x + 4, \\
p(x) &= 37137x^4 + 245922x^3 + 534667x^2 + 411094x + 103045, \\
N_1(x) &= 1379156769x^8 + 18265610628x^7 + 100192309254x^6 + 293537074164x^5 \\
&\quad + 495845231429x^4 + 490539243800x^3 + 279450495148x^2 + 84849357864x + 10641750132, \\
N_2(x) &= 1379156769x^8 + 18265610628x^7 + 100186664430x^6 + 293475629244x^5 \\
&\quad + 495587670117x^4 + 490018152864x^3 + 278924635092x^2 + 84595125192x + 10594761156. \\
x_0 &= 269037344, \\
p(x_0) &= 19456158510401119549889853567682124269 3, \\
N_1(x_0) &= 37854210398185391119909734603213541810938851919821474895863614596469844460 9 \\
&\quad 16 \\
&= 2^2 \cdot 7 \cdot 17 \cdot 7952565209702813260485238362019651640953540319290225818458742562283 \\
&\quad 5807691 \\
x_0 &= 272775528, \\
p(x_0) &= 2056025272032390385721219870528634952 21, \\
N_1(x_0) &= 4227239919235864887342627171906646939393534662984157984620794376712519022686 8 \\
&= 2^2 \cdot 3 \cdot 101 \cdot 348782171554114264632229964678766249124879097605953629094124948573 \\
&\quad 64018339.
\end{aligned}$$

$\underline{a = 24, b = 7, d = 21.}$

$$\begin{aligned}
c_1(x) &= -7x^2 - 43x - 15,\\
c_2(x) &= 7x^2 + 36x + 23,\\
c_3(x) &= x^2 + 8x + 7,\\
c_4(x) &= 2x^2 + 9x + 3,\\
p(x) &= 798x^4 + 8316x^3 + 26156x^2 + 22938x + 6281,\\
N_1(x) &= 636804x^8 + 13272336x^7 + 110934348x^6 + 472179624x^5 + 1078878880x^4\\
&\quad + 1312636376x^3 + 863392212x^2 + 292135424x + 40121116,\\
N_2(x) &= 636804x^8 + 13272336x^7 + 110867316x^6 + 471091656x^5 + 1072454392x^4\\
&\quad + 1296182464x^3 + 846125708x^2 + 284206952x + 38789332\\
x_0 &= 690918783,\\
p(x_0) &= 18184899087842619444275984674702427669,\\
N_1(x_0) &= 3306905548350193327319754018264349242277144657441520294326998521024598635 9\\
&\quad 248\\
&= 2^4 \cdot 2066815967718870829574846261415218276423215410900950183954374075640374 1\\
&\quad 47453.\\
x_0 &= 788336903,\\
p(x_0) &= 30821255426446056168210701558661649066 9,\\
N_2(x_0) &= 9499497860622304635852791775547287930524696647815446219666803926366390185 1\\
&\quad 088\\
&= 2^4 \cdot 5937186162888940397407994859717054956577935404884653887291752453978993\\
&\quad 865693.
\end{aligned}$$

$\underline{a = 29, b = 5, d = 37.}$

$$\begin{aligned}
c_1(x) &= -(1/2)x^2 + 24x + 23/2, \\
c_2(x) &= 14x^2 + 96x + 30, \\
c_3(x) &= 3x^2 + 8x + 3, \\
c_4(x) &= x^2 + 8x + 1, \\
p(x) &= (10045/4)x^4 + 31896x^3 + (228005/2)x^2 + 60120x + 35917/4, \\
N_1(x) &= (100902025/16)x^8 + 160197660x^7 + (6360030839/4)x^6 + 7575598140x^5 \\
&\quad + (135066868587/8)x^4 + 14293625076x^3 + (22676894823/4)x^2 + 1081276596x \\
&\quad + 1291797801/16, \\
N_2(x) &= (100902025/16)x^8 + 160197660x^7 + (6359428139/4)x^6 + 7573202220x^5 \\
&\quad + (134962592611/8)x^4 + 14267236308x^3 + (22615799123/4)x^2 + 1077869028x \\
&\quad + 1287774649/16 \\
x_0 &= 558110127, \\
p(x_0) &= 2436517704068701140989106431064374644 21, \\
N_1(x_0) &= 593661852224021470884231864299751692315609427590376956926324924374427918 46 \\
&\quad 592 \\
&= 2^6 \cdot 3^3 \cdot 343554312629642054909856402951245192312273974299986664887919516420 3 \\
&\quad 8652689 \\
x_0 &= 556747959, \\
p(x_0) &= 241281761119813502902407304706517574357, \\
N_2(x_0) &= 582168882490787469558544758565595174954798354544911806321200168199149682 58 \\
&\quad 608 \\
&= 2^4 \cdot 363855551556742168474090474103496984346748971590569878950750105124468 55 \\
&\quad 16163.
\end{aligned}$$

B. Pairing-friendly Genus 2 Curves: Numerical Data

B.1 Parameters produced by Algorithm 7

Here are some parameters found by Algorithm 7 for the CM field $K = \mathbb{Q}\left(i\sqrt{2-\sqrt{2}}\right)$ and embedding degree $k = 5$. Corresponding to this CM field there is a genus 2 curve defined over the rationals [59].

$$C : y^2 = -x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1.$$

The curves over prime fields corresponding to these parameters are either C reduced modulo p, or its quadratic twist C'.

On average, a MAGMA script found one set of parameters with $r = 160, 256, 512$ and 1024 bits in 0.0918, 0.3486, 2.9938, and 46.5615 seconds, respectively. The computations were performed on an AMD Quad-Core Opteron(TM) 2.4GHz computer running Linux kernel release 2.6.9-34.0.1.ELsmp; only one processor was used for computation.

$\underline{r : 160 \text{ bits. } k = 5}$.

$p =$ 252823257935282285362732638695054084330470208363294037922085422639242
9740214286170166852568584783960631710497763211466425437626783979662947366
79271737114219377482492730434694368080216503567747137

$r =$ 1461501637330902918203684832716283019655932544881

$N =$ 639195997530102770743719375835116542403184563967996666440138384615623
1104135942006766949461178052253303126123108270449109818252877992852236693
9854055782191379965677314562703378699008278543675026648680068400692359055
6954728131135395897277972576354640367835735384699586219721088378014250469
0516520543753456431447895666619342429338048350855555475511765095933553626
5110336972288875552378947584

$c_1 = 1$

$c_2 = 112432926212760798482063317306300237311742516999595699549737862$

$101371658215205518310568831884301922$

$c_3 = -6424814484839559442455782912278887167318368862383\,2$

$c_4 = -1098020179093273812297945051542599888895297113463\,80$

$\rho \approx 8.072$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = -x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1$.

r : 256 bits. $k = 5$.

$p = 70488107148090716207829667010286907438975845631687862097604525449948\,7$

$7530570186125117122017350141805247723779624730169393101671127446215490847$

$0128180097731192247524353202667866344441677798408664226182036087805320910$

$7260269920646366156330351242218700528276622717003991911130319025660067745$

840160149952389932917329

$r = 1157920892373161954235709850086879078532699846656405640394575840079 13$

129642241

$N = 4968573249320717521459123838938891694898356220337849895988806142299 69$

$9600573805281453411826215444363606741797229694154849558866843478727700264$

$1105324414001856604997470007681554137437103159261172089255501470358581691$

$0913734818476522890003367060634939104658599174570132609823174216276573137$

$8669572028319853268929729746434758497120580756345226145068054586116990212$

$0443929992312351457834418288528071757692892289663780177801079095634553929$

$6480701514721219823943376856364544844490404257431312550838391605233331165$

$2091324748046447124154493757683497657698145122503447211715505414438313883$

$5078630022905452819012061453102081426787555 2$

$c_1 = 1$

$c_2 = -5936670242993572074752240216934048675593535867493623642911929101631$

$1737731409117467973049416437737755512483626195984512654911475975189673396$

5375133869149502

$c_3 = -354880931356668387362428709913319025744571268059526422587605882999 0$

309058529874

$c_4 = -593697948081387184889577965812434116409665571501180864734898731859 6$

163181064168

$\rho \approx 8.093$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = 3(-x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1)$.

<u>r : 512 bits. $k = 5$</u>.

$p = 335008079246530563726120681491057120190326476848023623418849270705403$

025323869866804864722675993670155978032298911612298970772587807257095048 2

339382616758011777583609298242512899827689332995826994221172363916842412 2

835192981212836356652405731688402956757226538431141752940466421546103727 3

335386546360838507518135270947123160340531138589733038581083233829022937 3

629433864926251114506300179129297667736132975815746271616363329968809961 2

761859624081561517331937887287687178616501895576295485944805512184650359 2

520812773248391216120560540957577545455150053767296330120142373670676377 7

892828879742492044527038799989475169171601

$r = 134078079299425970995740249982058461274793658205923933777235614437 21$

764030073546976801874298166903427690031858186486050853753882811946569946 4

33649006084241

$N = 112230413160449702198126862108098653383586001483785667324770189028 68$

591298702406936591280416929517595313244047369787880933254744311895651104 8

924215827862280061644104999556329006841263664388618126753461769990647168 0

135684085159772998485886440912198167488523337353620591949482531790616511 4

886355056181650467284159308881991767975726409399306329201929244165981189 0

964429915584615514548193336400310989492893346850969715772399309737309942 4

731061707405024983533259659761705435304894764062784459326584661674404820 3

589990808281100547430010153769572974320399557923902098156706552989080432 8

586547274828963301478105441787187706209521580066135283247109339755099658 1

0829728607020925640934426760598315154081455815137896599996710082414116153
8758202861255529762349798235467307536842306680966547151560250464614627467
6280304178440896265086071976771393960186645433788202385227748354046664705
6043288274614079409654013832022798780433108257704454285600069637475601685
5668345721561150631035189424388958386770625722839174844558732827061335359
7396272574949736113573879375049271428943518090660991488549981999730343603
3701081705291652436301765100985664448658782887010301577613746326586686201
8085897066563266988620046862925185064744694134875859933038557645346837870
24780000

$c_1 = 1$

$c_2 = -40927257374916448444960043289401285052954569894041690241073368670$0
2808092727796979553126208534683675299755340836017188106288532761674422535
3601980879470278089965300447863592472705865956276784032006978786587497708
3277847405605138878998017257380919270399211683599748285124901246581378306
927829536072973884010608 50

$c_3 = 40718322247057160026195578357189524646988171776192499501124402253$11
7091727946183228269616280872992381120870809967134479817010530596677389181
2505702756277750

$c_4 = 2023129441092557194108329905958836658604534034683229765067161603442$04
4459739081672395276631140889276720174612101185345113541079509366213836026
00192437308270

$\rho \approx 8.0651$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = 11(-x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1)$.

$\underline{r : 1024 \text{ bits. } k = 5}$.

$p = 1049431317817967803513155537031147141647506147323473623570705557386$89
3013554313587137667850332005668823676933992122450896524826300645632284443
9709861761660664716766381039287716630666669243461281309213121059856150137
7408203980329843513963646096773624098217875284460094793183625579561324596

8608356367730215580187226775879901480970602565674238249031703056709423217
2131239853395900609409094397511164152363166477953010966113471921040846307
8296833046650930606393166184513292546742223326863687877494477464894473322
2580146352337166889019971413579722767523423370025599511111529312594779244
3541781393930719783466273872212843028790991003473843776986927819327031948
1713342165746135751191878684098557803535563450960706349481094799353674300
8633415032924583446378352447284142570110821165206981149264650477089508811
9114904402397502070681808025695867624940513358394564535019224685066746008
2416569948913907053025787910545410464358848282304832473112320828765889470
5604263949159057936370898664269118285653927024520802671750165732759812026
0688501468623837134188046046394520454894280067905925470874214764925409025
3851496741235700604148213969967287016951389613690467928986300565066816752
9053570183646180548072142460476120051703950764320138969111552605587351896
0657

$r =$ 1797693134862315907729305190789024733617976978942306572734300811577732
6758055009631327084773224075360211201138798713933576587897688144166224928
4743063947412437776789342486548527630221960124609411945308295208500576883
8150682342462881473913110540827237163350510684586298239947245938479716304
835356329624224142551

$N =$ 11013060908171565482778874116890142534684895971345437325047395973061
6759350987673635370386661323144384147441478939900055680762860287128384105
4902665836927719237593406042804513684315739066059397771282895303010787292
2179757197582159028597547853246866745923598008900586551168111551022146642
7797892056997175653567482674277676776365175778879398392915988749773939288
6435705233055514759060289001789676697626524970048461677042565997116692632
1574514938662017298145090019935086860014508760684579132668330742154526790
6729756575365103858396417152258336224360945964101533318634906150998015840
2926739626469233845687108138465424889309069890167784506422380877885738677
8486497648498874904198469516015126562709058573529499230570720716609495114

5176652748355956410449190437147438437365360933789948231562936734446248562

7333464760364261479389502053771882656875842881008586953138121719289691218

8758279287588811439673945582167757652834133685656149967100183974970665666

8301508675081476762833629781778446743401408792912743813248465789160097609

4907483265468514579887878917829916217551992851105645558151863186904311915

0897161484888708685607650964899796756659515412310704366815387550234691315

8131818826772780505198203396288812795508345351027967383176379701435096670

6087687519514932573232984872490858110150153420011230071225803878990363639

0228155715688332006641078957349065854370415790123363912464828804129733848

9175696250693127007079671001962884229986939989639986327089602241629610698

2541461860784718937644401952697781463951184575190864828437905119167020164

3344473177237375537182909982694143170480035384871367555930058478945286874

7897752470806228849738428129325218708334019884558804413088114397314208082

9684618582720600127298888204020537916245798945760355562052390503198184841

3388707887323066157837964264892073291036971698537389349632090868071247021

5487116535358735807812813655196187306147452925445662352702598796196616210

5611237005879204757511953065284647610819115873731988933920480554898200228

3914188267109522018962565363987515033850846271990633326448625370796130891

6187780991045712242162464946706937745089418294838706299085709554979040290

6303467139111643335427409219418781483564037022764043131466539004625154594

6209963899733654040406326188041008785111305118300000241509541850382046427

9921348250445947811452089231802857382987378154131554070179868476678463358

0643767635905430275544733949993084829695263010882447387128654441638838688

6796417377560269667075669849950440569287085578562873023654943240390761349

344

$c_1 = 1$

$c_2 =$ 7243725967407822239094060546538180769641539562317536530074567141903983

3083544386878828065677547992138075585030934397420775737590127163517898465

0846109282344565484417070274587173691100946634714875141158627632996168956

7766495789352183805458433940894170433679708077909086439456475244369359438
9518081325497632105610564738309170848248014811242603672360833841226847675
9481606098563108470451607635559144562515764965397762341417794410217622519
4338497201942951403070267420975047429409587019711286902398751160111773836
5807578038205778723458526333750943907335694407938046725757613069297767153
849328740268856946143191777694575094681 8

$c_3 = -1653510516671684447157854810693229731567170381931345156776401654 75$
6356287031722889250259390454834982137350158483356777003123281295331023971
1393360455936205111439243887010818642160393233292858255502407201894108071
9149943387259685239578041196541694819919059869390598882816649918129689175
47027543464862829090783742

$c_4 = -2073358691575217757388951192596202939719232864453686007577520507 86$
1398853142551817062560786443520326893830924572045474412814928410396990573
6933573731383048372745200257985204644144837697573959688350508891779753127
3479259087733008907698262801412853472379355142666972763160082464948044116
7446683063036638423281152

$\rho \approx 8.045$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = 5(-x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1)$.

B.2 Parameters produced by Algorithm 8

Below are some examples of the parameters found by Algorithm 8 for $K = \mathbb{Q}(i\sqrt{2-\sqrt{2}})$ and embedding degree $k = 3$. Here, we choose $C_3(x, y) = C_4(x, y) = xy$, $C_1(x, y) = x^2$ and $C_2(x, y) = -(a + b(1 + d)/2)y^2$ in Step 1 of Algorithm 8.

On average, our MAGMA implementation found one set of parameters with $r = 160, 256, 512$ and 1024 bits in 0.1092, 0.4468, 4.1718, and 50.0140 seconds, respectively. The computations were performed on an AMD Quad-Core Opteron(TM) 2.4GHz computer running Linux kernel release 2.6.9-34.0.1.ELsmp; only one processor was used for computation.

r : 160 bits. $k = 3$.

$p =$ 276032206782791857604308501919988591136740885931343898740256384866241
646755370297962312472363405383281006525389401749509877968225746849762659 6
054621968600128109029276968729859800558964868162387810481

$r =$ 1461501637330902918203684832716283019655932543447

$N =$ 761937791813779631994733941106633708154739036303135746201414612683681
374022951126862517606109944088144225942806086156441245392989328784595634 0
341615473801381877788622808833784218658203120398140352297108203162864445 0
834524316059579653777102002747137290912319563027848525351304927065061525 6
435136442386120895901675012299462125369911866209880438172735833621377815 6
291342604171682918546278978314937568

$c_1 =$ 853413751674246325960655910542033278192644078137851807206531855460335
897482560901762777003565546321

$c_2 = -$467312771754171603865894820458465529298297100229438686497717835334
951148694691783854304471959958498

$c_3 = c_4 = -$89309702244271126870314830090645570026648145619900427099516737
4051672546438742749426798352836518846

$\rho \approx 8.2401$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = 3(-x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1)$.

r : 256 bits. $k = 3$.

$p =$ 822920761971611209794051125149779261868007917105814333422807428702492
483230083267137722107007539895222282160142127021544643255654790661296929 3
703538932296757001914772160185501510936146565823839280291059897730788458 1
966993126278663824378978346229524223744879456228542389848372082725722442 1
582887155754347373346337

$r =$ 115792089237316195423570985008687907853269984665640564039457584007913
129640743

$N = 677198580483937194263730753359784807376570572162519246889869342280825$
3032215444487859365278749079347589549730845666733117453777198238279219494
5280678988988024443378725219717152986643553771096267443036427016707389095
7249248397038280644492111218229707870352901997265602267012008190367799204
2490892895555013596712575651692176016210908268738361775620639618631060792
5033229572686474111206272193416927126310352656009315433216497023049930883
5373318602217711383763542668793170469526104112283163915538814071400367342
3775883028281057290061738442630720051414075948315034087299281022702814170
14852155526683323382176465726972979082574048
$c_1 = 899567387391479217381476947274351584712780874649839002409060884043691$
7034478629557785770257234423972877031276763948663931761267676699233257997
62748414274889
$c_2 = -379916236281151103764633380973143102421074912906860994641809351833$
4237736166615736185164181781338965280295434753862169111244409012722954687
785372266393538
$c_3 = c_4 = 8267529934618186873729771614246762778267959823408343148411442228$
8087906493405752740627824201485645210824879536505195273507388849360615838
257032702979376742
$\rho = 8.0950$
The equation of the curve over $\mathbb{F}_p$ is $y^2 = -x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1$.

r : 512 bits. $k = 3$.
$p = 626094627977785411761504336964367530509161001161397733772648139606108$
7968126017678724733192354804393642278933710530797321862269310580976745634
6116911345645395717388268813930138137331860478986610924439831778326927171
6845062886926383345031759550556693174884602128438118374664571060501824065
9100037203110165020080488984639114234906464443216520747556803744300494494
6992418036349155347397250087843163119953929796859079789902695120224849161
7516608669145239915106433027259815435477015548796801852943295822260576268

0312122940392880866229093627162407463271989320163774597151730679749981207452016538398720367688045883149174081800488l

$r =$ 13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006090369

$N =$ 39199448318264151528178525307479058638395060577363482831228191455392430617072216737586391392823631454088559653178876314058058357499296345711390330645883769407087588861368306783250067815987192235846514496539626229241512872125633387397232888685245262659279556319304400180637375517474612769749137762505901562356541685690314600554383215073125331569586937512645166120298264279408059464649976075153909251124903466418613010749549204730675846530541334466867848900995961007749305966157811682822427733338186419810693149696001286114555979086027215835151893784200930042730938843980293629259932341674782220853825107820445471454896497607745368481116313557548784332268779672950773243168551391982315656665946290659318848645120038715037490277601059649289724286892606151320217014818084746352778138433559279734352470295011903576974979898694593744266772555427406571416731601344160182507921496655124334059849102282719924031318729315786963545054984777135878751905642733266163172345242410605716052606453741536013111869784525608250539564504087388524613098970682086686742156827007879617577585219269695723764760354784310005388049039328216764676942948605157384040705803125562906559984920395123564544004540658629004128489598913587376698221064589729960790460616492155655331842

$c_1 =$ 17402662210303604900144320367278731716570996082548387954776152870110535129890690469383994201632346156643913774653209882482781725799537133793860373544454124284273225423842889374337640211063105958131440397513480175766337233403312989928095855456486957655912332828441426398809429310462548028801764401665005364985104

$c_2 = -$414898830634725511579876261796925209423483197180808499033365137192
7558171176108301510977080798335020411125167778584725421392063859461695138
4087271477429444214749212952259453844250997349939835035354149654600838848
4697319851167904916270475649949946784682228312403248649006236852585156268
41625371289867006618563072

$c_3 = c_4 = -$12016941541828430890387541295049543238634890141937497877421825
3608511313770187340039750897734731896175653227833547978740517617931580691
9494419145235744346056001344596541225696787382859793476317869029518461205
0052095264627090978218925706008532351098389876395449170546152881538676448
6819410816179248899825935663584

$\rho = 8.0816$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = 11(-x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1)$.

$\underline{r : 1024 \text{ bits. } k = 3}$.

$p =$ 717748645781505731120599269684910693590944029612987068440967012314130
6637827393231944795966268238394707373418250637693458042904697055796191300
6988518257403915714005639997828602961606379818049724696348918738839210978
4884097588161710751447842237401094267825596641326555653496480397321236084
3359687016377451991881531057275736765140557796850130723269247861970378423
4747661610312631323994375446803092065579965869275407690364653384443149636
1358359960327719785761624131833934060542855381519429968298504355243330594
8128051188598482445090960279949827080200814185109371664267078220279517108
4593441499198308986503899040312785820990605173937322268693944141897213905
0350040373032830045642801185170518236713373491710503302901558703113134368
3241826353657585244175657105742633183737223279548444715679485406276250355
1201137915874208065998822573397075248659377281765538966698467194444038688
2544976363374550235175548720819849709645412976171386385886955140368425643
5595018639856563961113295199752946528182355932639678152695494113031640748
6915218894642959537179051946992663567484366871922820680694703814552225520

4643739689679297558549156573384073054580960265102343844077312858660618523
0160019449034269795423287562037274184579733091032396846997359646292442899
233

$r =$ 1797693134862315907729305190789024733617976978942306572734300811577326
758055009631327084773224075360211201138798713933576587897688144166224928
4743063947412437776789342486548527630221960124609411945308295208500576883
8150682342462881473913110540827237163350510684586298239947245938479716304
835356329624224143009

$N =$ 5151631185211853847538401225830108563827092494286719626614232113650503
1293987383778371298593754137105995138948867912378299794411476758795871018
1278043789858785925791814007167960522894003112009473219636386458462918330
8572012795702329118286311261069839092889921336158767998376066369194229567
3513260745550660874762195725201585184123028775422412983573938987294941971
3196022577704427384780141919851463718623331917109357438812637652672814769
4916765019526622151233984195487133501965159128998750893140803288656210948
3688152105838154819768965300636443948244526903932740331838170510913212334
5580157341139423056241213023888027824520742343327842656474573445288152394
1452623676380337409145257444921641748939607933660774575746247498959041443
2242125178866814444977581681308130715131198922885500625566085495227779556
1073237881267963469150490201777657940835109165764630357927788127673388742
5601581694651165498847639765392244601101123798593305363230649856451002458
9188893662158090013449845706382989980910575944404135904492207231887797647
1499200485710259642399365873681714674442077429607130873478840789896098741
7154345683466548989502357105930894117350277410693481985204479718575323399
1660051685402203896513522323157037222177587412610583259728490303909719126
7789015535251885406537263099073247680614736793929707306447512293674518982
8617758064974025295374374751584040562659521290518541217903704812257538386
9579270049956132830108339126836163342173596962311896894993036112669553313
939555492471860484630000856664302452824150794096867637988697308682265590

6133242863267943605360732832950584508750182506668160375930488838808196330
8321237485300246428442097317214290631861287510288955910881362062874360853
5360183755807572962437618631563317241283618789061604351451958568659453355
5085328642120047173695576212225431262429587305870554292547099393283843789
8778772339629902350231517065379800315959172287505333819272066820058506445
1486781599426285596652733281850226828122246033844848187647739088611155954
4990708805981662320939385423098611058587644744015264671940931506576574796
6574945135893523272284538857096770034016880047082680580540618565741917115
6981433598158488000847546249792572546753218368775523920043139133607183248
7570805707999371196914500444836433288499776969698483926470544505629385606
7742312323325978499157829066471585493650335737156015959885576560769234421
7163798948736397105796266249161358253213474072307345140179161963167234174
9770068842895220487754663826788480959975507429493447607951826211093618442
24

$c_1 = 2410468099113317448297342148494953382883830334225165623439529111 53215$
8192841241473577315249020488024863361377610028395777932327114358923433565
0936466052461749239726108485714966836992014266020206655946581519964385429
0867445800142955439023414996484770332209901425758580132297217861282561935
4736005546869003077727086069051848857786861113120469969485021821036783977
1394120658420848096491899664988059309107088706236032356462445760157820395
8309864823697346438670189432998903144987790168082978899647052970353731903
1854636379409714950811422783948892128814057829596138961269831112117378178
58343235886567269468918151971418107870 25

$c_2 = -381790331864082701677964116490390471323211246420800974547315434670$
6247391478402571173301059272864468354483707354443499461488099833644412239
0172406114284649073743487814335212519672617012878831879092793266950084898
9331284076284350734797021226156120050658417295790918253642974591025047786
9753645596675223611665212263778651718415338373723285748183989896258950882
3445742838796314746308762938191147456388442030672510020289063226826761151

008193655722354022176235389240443758003210236100654445066798360110473882193898992803593435001301393173824785916438956056680469282803585359156883628782859425827958436920858894640806217611552

$c_3 = c_4 =$ 429020609180551619449336320764971942330111267150387784558918306394586501434628222106634617937244089757608691079278986899772953479194078187223733683536372806790780598134032914251925948674203980107215348783360020578780755493255263900898034212762065715884084766268514265114974272284259745771996026235812265475888681834988691541451183610643465792889944147364509265001746418610956580798299508248257025740938108042076570815603994956671231242513691501293706280383000547033138013142181563173275855935787068125676815147103415272424675034157667902311183100298814367256007564896238960083760308633592317742627253038661001979530245160

$\rho = 8.0444$

The equation of the curve over $\mathbb{F}_p$ is $y^2 = -x^5 + 3x^4 + 2x^3 - 6x^2 - 3x + 1$.

C. Some Source Code

C.1 PARI/GP scripts for finding parameters for cryptographically-strong genus 2 curves

Code for case $d \equiv 2, 3 \pmod 4$.

```
\\ Timing for generating suitable pairs (p, N) for genus 2
\\ cryptography.
\\
\\ Script outputs results for generating $count (p, N) in
\\ $time seconds. It returns either when $count_max pairs
\\ are generated or when $time_max has been reached.
\\
\\ d = 2, 3 (mod 4)
\\-------------------------------------------

allocatemem(40*10^6);
default(debugmem,0);
default(timer,0);

prmbitsize = 400;
MAX_COFACTOR = 2000;
\\---------------- D = 0 mod 4 ---------------

\\ Set count_max and time_max
count_max = 500;
time_max = 600;
outfile = "search_result_D0.txt";
{
\\ Generate good a, b, d for CM field K = Q(i*sqrt(a + b*sqrt(d)))
Good_abd = 0;
while (!Good_abd,
d = random(20) + 1;
 if (Mod(d, 4) != 1 && issquarefree(d) && qfbclassno(4*d) == 1,
  a = (random(20) + ceil(sqrt(d)))*(-1)^random(2);
  b = random(ceil(sqrt(a^2/d))); \\ This ensures a^2 - b^2*d >= 0.
   if ( !issquare(a^2 - b^2*d) && issquarefree(a^2 - b^2*d) && 1+d+a+a*d+2*b*d >
 0, Good_abd = 1);
```

```
 ); \\ end if
); \\ end while
}

\\------------Polynomial to be factors ------------
{
f(x, y) = 2*x*y*a + x^2*b + y^2*b*d;
}

\\------------- p(x) ----------------------------
{
p(x1, x2, x3, x4) = x1^2 + x2^2*d + x3^2*a + x4^2*a*d + 2*x3*x4*b*d;
}

\\---------- Possible group orders ---------------
{
N1(x, x1, x2) = (x + 1)^2 - 4*(x + 1)*x1 + 4*(x1^2 - x2^2*d);
}

{
N2(x, x1, x2) = (x + 1)^2 + 4*(x + 1)*x1 + 4*(x1^2 - x2^2*d);
}

\\------------- Old Method --------------------
\\ Takes on input [count_max, time_max].
\\ Function returns if either count reaches count_max or time reaches time_max.
\\ Returns [count, time] - count primes found in time seconds.
\\
Timing_DO_old(count_max, time_max) =
{

local(OK, count, prm, c1, c2, c3, c4, time_start, time_end, fact, factN, factsiz
e, myN);
local(idx);

OK = 1;
prm = 4; \\ isprime(p) == 0
count = 0;
time_start = gettime();
time_end = 0;
while ( OK,
   c3 = random(2*2^(prmbitsize/4)) - 2^(prmbitsize/4);
   c4 = random(2*2^(prmbitsize/4)) - 2^(prmbitsize/4);
   while (gcd(c3, c4) != 1, c4 = random(2*2^(prmbitsize/4)) - 2^(prmbitsize/4));
```

```
   if (Mod(c3^2*b - c4^2*b*d, 2) == 0,
     n = (1/2)*(-2*c3*c4*a-c3^2*b-c4^2*b*d);
     fact = factorint(n);
     c1 = prod(row = 1, matsize(fact)[1], fact[row, 1]^random(fact[row, 2]+1))*(
-1)^random(2);
     c2 = n/c1;
     prm = c1^2+c2^2*d+c3^2*a+c4^2*d*a+2*c3*c4*b*d;

     if ( isprime(prm),
\\ && prm >= 2^(prmbitsize-1) && prm < 2^prmbitsize,
       myN = [(prm+1)^2-4*(prm+1)*c1+4*(c1^2-c2^2*d), (prm+1)^2+4*(prm+1)*c1+4*(
c1^2-c2^2*d)];

       for (idx = 1,2,
        for (cofactor=1, MAX_COFACTOR,
         if (myN[idx] % cofactor == 0, max_cofactor_test = cofactor);
        ); \\ end for cofactor

        if (isprime(myN[idx]/max_cofactor_test),
          write(outfile, "[p, N] = [", prm, ", ", myN[idx], "]");
          count++;
          time_end += gettime();
         ); \\ end if isprime
       ); \\ end for idx
     );\\ end if
   ); \\ end if


 OK = OK && count < count_max && time_end/1000 < time_max;
); \\ end while

return ([count, floor(time_end/1000)]);
}

\\------------ New Method -----------------
\\ Takes on input [count_max, time_max].
\\ Function returns if either count reaches count_max or time reaches time_max.
\\ Returns [count, time] - count primes found in time seconds.
\\
Timing_DO_new(count_max, time_max) =
{
local(OK, count, upper, a2, a1, a0, b2, b1, b0, myf, myc1, myc2, \
prm, c1, c2,c3, c4, ppoly, fact, factN, factsize, idx, i, j);
```

```
local(time_start, time_end);

OK = 1;
polyfound = 0;
count = 0;
upper = 10;
for (a2 = 1, upper,
  for (a1 = 1, upper,
    for ( a0 = 1, upper,
      for (b2 = 1, upper,
        for (b1 = 1, upper,
          for ( b0 = 1, upper,
            if (!polyfound && gcd(a2*x^2 + a1*x + a0, b2*x^2 + b1*x + b0) == 1,
              myf = f(a2*x^2 + a1*x + a0, b2*x^2 + b1*x + b0);
              fact = factor(myf);
              if (OK && poldegree(myf) == 4 && matsize(fact)[1] == 2,
                c3 = a2*x^2 + a1*x + a0;
                c4 = b2*x^2 + b1*x + b0;
                c1 = -1/2*polcoeff(myf,4)/(polcoeff(fact[1,1]^fact[1,2],2) \
*polcoeff(fact[2,1]^fact[2,2], 2))*fact[1,1]^fact[1,2];
                c2 = fact[2,1]^fact[2,2];

                ppoly = p(c1, c2, c3, c4);
                if (OK && polisirreducible(ppoly) && numerator(gcd(ppoly)) == 1,
                   for ( j = 1, 100000,
\\print("j = ", j);
                    value = random(2^(prmbitsize/4) - 2^(prmbitsize/4-1));

                    myc1 = subst(c1, x, value);
                    if (myc1 == round(myc1),
                      prm = subst(ppoly, x, value);
                      if (isprime(prm), polyfound = 1; break(7);)
                    );
                   ); \\ end for j
                ); \\ end if OK
               ); \\ end if OK
            ); \\ end if polyfound
        ); \\ end for b0
       );  \\ end for b1
      ); \\ end for b2
     ); \\ end for a0
    ); \\ end for a1
   ); \\ end for a2
```

```
\\ Now polynomials are found
print("Poly found!");

 gettime();
 time_end = 0;

 while(1,
  value = random(2^(prmbitsize/4) - 2^(prmbitsize/4 - 1));
  myc1 = subst(c1, x, value);
  if (myc1 == round(myc1),
   myc2 = subst(c2, x, value);
   prm = subst(ppoly, x, value);

   if (isprime(prm),
    myN = [N1(prm, myc1, myc2), N2(prm, myc1, myc2)];

    for (idx = 1,2,
     for (cofactor=1, MAX_COFACTOR,
      if (myN[idx] % cofactor == 0, max_cofactor_test = cofactor);
      ); \\ end for cofactor

     if (isprime(myN[idx]/max_cofactor_test),
  \\    write(outfile, "[c1, c2, c3, c4] = [", c1, ", ", c2, ", ", c3, ", ", c4,
 "]");
      write(outfile, "[value, p, N] = [", value, ", ", prm, ", ", myN[idx], "]")
;
      count++;
     ); \\ end if isprime
    ); \\ end for idx


   ); \\ end if isprime

    time_end += gettime();

    OK = OK && (count < count_max) && (time_end/1000 < time_max);
    if (!OK, return ([count, floor(time_end/1000)]));
  ); \\ end if myc1
 ); \\ end while 1

}

\\------------ Benchmarking -------------
{
```

```
print("[a, b, d] = [", a, ", ", b, ", ", d, "]");
write(outfile, "[a, b, d] = [", a, ", ", b, ", ", d, "]");
print("Searching for suitable parameters...");


write(outfile, "NEW METHOD");
Time_new = Timing_DO_new(count_max, time_max);
print("New method finds ", Time_new[1], " suitable pair(s) (p, N) in ", Time_new
[2], " seconds");

write(outfile, "OLD METHOD");
Time_old = Timing_DO_old(count_max, time_max);
print("Old method finds ", Time_old[1], " suitable pair(s) (p, N) in ", Time_old
[2], " seconds");


}

\\-------------- EOF ----------------------
```

C.2 MAGMA scripts for finding parameters for pairing-friendly genus 2 curves

```
‘‘PFfinder1.txt’’
// Magma script finding parameters of PF genus 2 curves
// rho =~ 8
// Set c1 = +/-1
// Embedding degree k = 5

// Start of timing
tstart := Cputime();

para_count := 0;

bitsize := 160; // Bit size of r
// For Windows users, the following line may not work.
// Change to outfile := "file_name_you_prefer" should do.
outfile := "result." * IntegerToString(bitsize) * "bit." * \
 Pipe("tr -d \"\n\"", Pipe("date \"+%Y.%m.%d.%H.%M.%S.txt\"",""));
a := 2;
b := -1;
d := 2;
```

```
//a := 24;
//b := 7;
//d := 21;
c1 := 1;
k := 5;
fprintf outfile, "bitsize = %o, [a, b, d, k] = [%o, %o, %o, %o]\n", bitsize, a, b, d, k;

twiceC2 := function(x3, x4)
    return -(b*x3^2 + 2*a*x3*x4 + b*d*x4^2)*c1;
end function;

NeededPrime := function(x2, x3, x4)
    return (a*x3^2 + 2*b*d*x3*x4 + a*d*x4^2 + 1 + x2^2*d);
end function;

r:= NextPrime(2^bitsize: Proof := false);
while  r mod k ne 1  do
        r := NextPrime(r + 1: Proof := false);
end while;
solution_found := false;

while not solution_found do
P<c2,c3,c4,p> := PolynomialRing(GF(r),4);

D1 := a^2 - b^2*d;
D2 := -d*(a^2 - b^2*d);
dr := Sqrt(GF(r)!4);

I := ideal<P | \
b*c3^2 + 2*a*c3*c4 + b*d*c4^2 + 2*c1*c2,\
c1^2 + d*c2^2 + a*c3^2 + 2*b*d*c3*c4 +a*d*c4^2 - p,\
(p+1)^2 - 4*(p+1)*c1 + 4*(c1^2-c2^2*d),\
p^4 + p^3 + p^2 + p + 1>;

Solution_over_GFr := Variety(I);

if not IsEmpty(Solution_over_GFr) then
        para := Solution_over_GFr[1];
        solution_found := true;
else
solution_found := false;
r := NextPrime(r+1: Proof := false);
end if;
```

```
end while;

CC2 := Integers()!para[1];
CC3 := Integers()!para[2];
CC4 := Integers()!para[3];
PP := Integers()!para[4];

  for i:= -100 to 100 do
    cc3 := CC3 + i*r;
    for j:= -100 to 100 do
      cc4 := CC4 + j*r;
      cc2 := twiceC2(cc3, cc4);
      cc1 := c1;
      if IsEven(cc2) then
        cc2 := (cc2 div 2);
        pp := NeededPrime(cc2, cc3, cc4);
        if IsPrime(pp: Proof := false) then
                N := (pp+1)^2 - 4*(pp+1)*cc1 + 4*(c1^2-cc2^2*d);
                fprintf outfile, "p = %o\n", pp;
                fprintf outfile, "r = %o\n", r;
                fprintf outfile, "N = %o\n", N;
                fprintf outfile, "c1 = %o\n c2 = %o\n c3 = %o\n c4 = %o\n\n",
cc1, cc2, cc3, cc4;
                fprintf outfile, "rho = %o\n", 2*Log(pp)/Log(r);
                para_count := para_count+1;
        end if;
      end if;
    end for;
  end for;
//CC2;

// Cputime
tspent := Cputime(tstart);
fprintf outfile, "\nCputime: %o, pairs found: %o\n", tspent, para_count;
```

VITA

VITA

Ning Shang was born in Anyang, Henan, China. He pursued his undergraduate studies at Wuhan University, China. He earned his Bachelor of Science degree in mathematics in June 2002.

Ning Shang started his graduate studies at Purdue University, USA, in August 2002. He received a Master of Science degree in electrical and computer engineering in December 2007. He earned the degree of Doctor of Philosophy in mathematics in May 2009.