NAIST-IS-DD1361001

Doctoral Dissertation

Pairings on Hyperelliptic Curves of Genus 2 at High Security Levels

Masahiro Ishii

September 16, 2016

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Masahiro Ishii

Thesis Committee:

Professor Kazutoshi Fujikawa	(Supervisor)
Professor Minoru Ito	(Co-supervisor)
Professor Shoji Kasahara	(Co-supervisor)
Associate Professor Ismail Arai	(Co-supervisor)
Professor Tsuyoshi Takagi	(Kyushu University)
Professor Atsuo Inomata	(Tokyo Denki University)

Pairings on Hyperelliptic Curves of Genus 2 at High Security Levels*

Masahiro Ishii

Abstract

Pairings on hyperelliptic curves including elliptic curves have been applied to many cryptographic schemes (e.g., functional encryption and its varieties), and the various optimization methods that increase the speed of pairing computations have been exploited. In contrast to pairings on elliptic curves, hyperelliptic curves are not considered to be more efficient than elliptic curves for constructing general pairings. However, the extent of the difference between pairings on elliptic curves and pairings on genus 2 hyperelliptic curves has not been clarified.

This dissertation is aimed at clarifying this difference by first examining suitable pairing-friendly genus 2 curves for pairing purposes. Other researchers have proposed the construction of genus 2 curves, we show that the Kawazoe-Takahashi families of curves are the most efficient for constructing pairing at 192- and 256-bit security levels and also provide the cost estimations of several pairings on the curves.

Furthermore, we exploit a new variant of Weil pairing with an automorphism and propose an effective calculation method.

This is the first detailed evaluation of pairings on genus 2 hyperelliptic curves at the high security levels.

Keywords:

Optimal pairings, Weil pairings, Efficiently computable automorphisms, Pairing-friendly Hyperelliptic curves, Kawazoe-Takahashi curves, Twists

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1361001, September 16, 2016.

Acknowledgements

It was impossible to complete this dissertation without supports from professors, colleagues, friends, and family. I am very grateful to all the people.

Firstly, I would like to thank my supervisor Professor Kazutoshi Fujikawa for his guidance and help. He gave me opportunities to tackle research topics as I want, and produce research findings. I also thank to Professor Atsuo Inomata, who has been supervising me for the cryptographic research. Many useful advice was a great help to perform the research.

I would like to express my gratitude to professor Hideki Sunahara and Suguru Yamaguchi, and associate professor Satoshi Matsuura. They provided insightful comments and their comprehensive knowledge not only about cryptology but also about information security. The helpful advice and comments from the jury consisted of the following members: professor Kazutoshi Fujikawa, Minoru Ito, Shoji Kasahara, associate professor Ismail Arai, professor Tsuyoshi Takagi, and Atsuo Inomata have enhanced the research. Especially, I would like to thank to Professor Tsuyoshi Takagi, who is an expert in this research. The discussions about my research with him enable me to obtain the good research findings.

Most of my research findings were based on the research activities when I stayed at INRIA Nancy – Grand Est, LORIA, CARAMEL (presently, CARAMBA) project-team in France. I would like to express my heartfelt gratitude to Dr. Jérémie Detrey and Dr. Pierrick Gaudry, who has been supervising me at LORIA. They are experts in cryptographic pairing and computational algebra including the theory of hyperelliptic curves in both theoretical and practical aspects. They advised me on not only technical matters but also basic and essential things about the research. I thank all the people who gave me the chance to visit to LORIA and join in the research project.

I am deeply grateful to all inet-lab staffs and colleagues. I have tackled my research in the great environment without stress. I used to play tennis with associate professor Satoshi Matsuura and other members. Innumerable discussions and talks with the members always encouraged me. I had a wonderful time with inet-lab members.

Finally, I would like to express the deepest appreciation to my parents. Although I spent my university life for a long time and devoted myself to research,

they have always encouraged me and supported my life. Without their kind help, this dissertation has never been completed.

Contents

List of Tables

List of Notations

$a \mid b$	a divides b
$a \nmid b$	b is not divisible by a
$\mathbb{F}_q$	the unique finite field of order q
C	a hyperelliptic curve
g	genus
Jac_C	the Jacobian variety of C
$\overline{D} \in \mathrm{Jac}_C$	divisor class of a divisor D on C
$\rho(D)$	a unique reduced divisor which is equivalent to a divisor D
$\epsilon(D)$	effective part of the reduced divisor D
$[i]D$	a unique reduced divisor $\rho(iD)$
$\deg(D)$	degree of a divisor D
(f) or $\mathrm{div}(f)$	principal divisor of the rational function f
$f_{i,D}$	the Miller function which is a rational function with divisor $(f_{i,D}) = iD - \rho(iD)$
$g_{[a]D,[b]D}$	the normalized rational function with the divisor $[a]D + [b]D - [a+b]D$
C_t	a twist of C
π	the q-th power Frobenius morphism
k	the embedding degree of Jac_C with respect to r where r is a prime order of Jac_C
ρ-value	a parameter defined by $\rho = (g \log q)/\log r$
μ_r	the group of r-th roots of unity
$t(\overline{D_1}, \overline{D_2})$	(modified) Tate-Lichtenbaum pairing
$e(\overline{D_1}, \overline{D_2})$	Weil pairing
$a_{s,h}(\overline{D_1}, \overline{D_2})$	HV (Hess-Vercauteren) pairing
$a_{s_e,h}(\overline{D_1}, \overline{D_2})$	twisted HV pairing
$e_{s_e,h}(\overline{D_1}, \overline{D_2})$	Weil-type HV pairing
P_∞	the infinity point on C
$\mathrm{lc}_\infty(f)$	the leading coefficient of a rational function f at P_∞
$\varphi(k)$	the Euler totient function
Φ_n	the n-th cyclotomic polynomial

1. Introduction

Pairing-based cryptosystems have now become a major research topic in cryptography and have attracted more attention because of the increasing interest in the efficient cryptographic protocols with pairings, e.g., ID-based cryptography. Modern public-key cryptosystems which are widely used in the implementation of SSL/TLS today are based on the Diffie–Hellman key exchange protocol [29] on a multiplicative cyclic group in a finite field or an elliptic curve group, or the RSA cryptosystem [75].

The traditional cryptosystems, such as RSA and elliptic curve cryptography run fast and effectively enough to establish secure communication in the Internet. Such cryptosystems, however, have limited functionality. Instead, pairing-based cryptosystems are more costly since a cryptographic pairing itself has more complicated mathematical structure and its calculation cost is considerably high. Thanks to the mathematical features, pairings can be utilized to design efficient cryptographic protocols which have not been provided by the traditional primitives. Some pairing-based cryptosystems can be utilized in cloud computing with ensuring the security of cloud services.

This dissertation explores cryptographic pairings that are used to build pairing-based cryptosystems. Especially, the pairings on hyperelliptic curves of genus 2 are analyzed to clarify the most efficient construction of pairings and their detailed calculation cost. The main contribution of this dissertation is to reveal the extent of the difference in the performance of a pairing between the elliptic and genus 2 cases.

The next section describes a short history of cryptographic pairings and basic pairing-based cryptographic schemes. In Section 1.2, the motivations and objectives of this research are given. The detailed contribution of this research are showed in Section 1.3 with describing the organization of this dissertation.

1.1 Cryptographic Pairings

1.1.1 Hyperelliptic Curve Cryptography and Pairings

Elliptic curve cryptography (ECC) was proposed independently by N. Koblitz [60] and V. Miller [68]. The fundamental building block for ECC is the elliptic curve

discrete logarithm problem (ECDLP) which is the following computational problem. For an elliptic curve E defined over a finite field $\mathbb{F}_q$ and points $P, Q \in E(\mathbb{F}_q)$, find an integer a satisfying $Q = aP$, if any.

Koblitz [61] suggested a hyperelliptic cryptosystem using Jacobians of hyperelliptic curves as arithmetic generalizations on groups of elliptic curves. As in the case of using an elliptic curve, the discrete logarithm problem in Jacobian of a hyperelliptic curve which is called hyperelliptic curve discrete logarithm problem (HCDLP) can be used for designing some cryptographic protocols with hyperelliptic curves. Arithmetic on Jacobians of hyperelliptic curves is more complex than on elliptic curve groups. Alternatively, we can use smaller finite fields, i.e., we can employ smaller size keys by using higher genus curves to achieve the same level of security.

Pairings have been studied in the mathematical research areas e.g., algebraic number theory and algebraic geometry [73]. The pairing on elliptic curves can be found in [84]. Especially, the Weil pairing and its detailed properties were described in [84, Section III.8]. For an elliptic curve E and the Weil pairing e over $E[r]$ which is the r-torsion subgroup of E, the Weil pairing has the following basic properties [84, Proposition 8.1]:

1. It is bilinear:

$$e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q),$$
$$e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2).$$

2. It is alternating:

$$e(P, P) = 1.$$

3. It is non-degenerate:

   If $e(P, Q) = 1$ for all $P \in E[r]$, then $Q = O$ where O is the point at infinity.

The bilinearity is the most important feature for building applications with cryptographic pairings.

Pairings on elliptic curves or higher genus curves have attracted significant attention and have been applied to many cryptographic schemes, such as ID-based cryptography. Generally, calculation methods for pairings are complex and

the cost of pairings is considerably higher than that of arithmetic on curves. In addition, the cost is considerably higher when using algebraic curves of higher genus.

1.1.2 Public-Key Cryptosystems Using Pairings

In the literature, pairings were first used to attack elliptic curve cryptosystems on supersingular curves [66]. In addition, many cryptographic schemes with pairings have designed so far. Some applications of pairings were initiated by Sakai et al. [77] that constructed the efficient identity-based encryption firstly, by Verheul [91] that used the Weil pairing on supersingular elliptic curves for the certificate systems, and by Joux [52] that offered the protocol for three-party Diffie–Hellman key exchange. Indeed, the three-party Diffie–Hellman key exchange protocol can be designed with a pairing e as follows:

> Suppose that three people, Alice, Bob, and Carl want to share a secret value for the key. At first Alice, Bob, and Carl agree to use an elliptic curve E and points P, Q on E. Alice chooses a secret integer a and computes (aP, aQ), and publish (aP, aQ). Similarly, Bob and Carl publish (bP, bQ) and (cP, cQ), respectively.
>
> In order to compute the shared value, Alice computes the pairing
>
> $$e(bP, cQ)^a = e(P, Q)^{abc}$$
>
> with her secret integer a. Bob can obtain the shared value with computing
>
> $$e(aP, cQ)^b = e(P, Q)^{abc},$$
>
> and Carl computes the value
>
> $$e(aP, bQ)^c = e(P, Q)^{abc}.$$
>
> Consequently, Alice, Bob, and Carl shared the secret value $e(P, Q)^{abc}$ by using the pairing e.

Note that symmetric pairings $\hat{e}$ with distortion maps ϕ where $\hat{e}(P, \phi(P)) \neq 1$ have been considered for designing pairing-based protocols. However such pairings are not appropriate to use or inefficient at high security levels at the present time.

By using the bilinearity of the above Weil pairing e, we can consider the bilinear Diffie–Hellman (BDH) problem:

given $P, g \in E[r],\ g^a, g^b$, where $1 \leqslant a, b < r$, to compute $e(P, g)^{ab}$.

Many pairing-based cryptosystems work under the BDH assumption which means no polynomial time algorithm can achieve non-negligible advantage in computing the BDH problem.

Identity-based (ID-based) cryptography is the fundamental pairing-based protocol based on BDH assumption. In ID-based cryptosystems, a user's public key can be chosen by themselves where the key identifies the user, for instance their email address is used. Practical ID-based cryptosystems were offered by [19, 77], here is the basic system by Boneh and Franklin in 2001 [19].

Setup:

A trusted authority (key generation center) publishes an elliptic curve E, the modified Weil pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ on E, a cryptographic hash function $H_1\colon \{\mathbf{ID}\} \to \mathbb{G}_1^*$, and a cryptographic hash function $H_2\colon \mathbb{G}_2 \to \{0,1\}^n$ for some n.

Pick a random integer s (master secret key) and publish P and sP where P is a random generator in $\mathbb{G}_1$.

Private key extraction:

A user chooses a public key **ID**.

The trusted authority computes $Q_{\mathbf{ID}} = H_1(\mathbf{ID}) \in \mathbb{G}_1$, and set the private key $d_{\mathbf{ID}} = sQ_{\mathbf{ID}}$.

Encryption:

In order to encrypt a plain text m under the public key **ID**, compute $Q_{\mathbf{ID}} = H_1(\mathbf{ID})$, choose a random integer r, and set the cipher text to be

$$C = (rP,\ m \oplus H_2(\hat{e}(Q_{\mathbf{ID}}, sP)^r)).$$

Decryption:

For a cipher text (U, V) encrypted using the public key **ID**, compute

$$V \oplus H_2(\hat{e}(d_{\mathbf{ID}}, U)) = m$$

to decrypt.

Many other pairing-based cryptosystems have been developed: proxy re-encryption [6, 65], broadcast encryption [20], short signature [21], attribute-based encryption [43, 76], functional encryption [22].

1.2 Motivations and Objectives of This Research

Pairing-based cryptography has attracted attention since there are many useful cryptographic protocols with pairings as described in the previous section. In the literature, pairings on hyperelliptic curves are not considered to be more efficient than elliptic curves for constructing general pairings. In contrast to pairings on elliptic curves, hyperelliptic curves are not considered to be more efficient than elliptic curves for constructing general pairings [38]. Hyperelliptic curves, however, have some features that can be considered advantageous for pairings listed in [7] as open problems as an extension of those in [38]. For instance, the automorphism groups of genus 2 curves are larger in size, and have twists of a higher degree. Therefore, construction of the most efficient pairing for a given genus 2 curve and the extent of the difference between the elliptic and genus 2 cases remain unclear.

This dissertation is aimed at clarifying this difference by first examining suitable pairing-friendly genus 2 curves for pairing purposes. Other researchers have proposed the construction of genus 2 curves [37, 47, 54, 58]. Our work focuses on the Kawazoe-Takahashi families [58] of curves for which we evaluate the costs in detail.

Scott [79] noted that the Weil pairing may become more efficient than the Tate pairing at a high security level since the cost of arithmetic in the extension field is much higher and the final exponentiation is unnecessary for Weil pairing. We undertook to compare different methods for the computation of various pairings on genus 2 curves by evaluating a Weil-type pairing with an efficiently computable automorphism in addition to Tate-type pairings. Our research led to the proposal

of a new variant of Weil pairing with an automorphism and in this paper an effective calculation method for the pairing was provided.

1.3 Contributions

This dissertation comprehensively explores pairings on hyperelliptic curves of genus 2. To reveal the state-of-the-art results of previous works on pairings, Chapter 2 first reviews the definitions and fundamental properties of pairings, and introduce the state-of-the-art pairing-friendly curves of genus 1 and 2.

The main contributions are outlined with describing the organization of this dissertation.

1. A selection of appropriate curves for an effective pairing is revealed in Chapter 3. The selection is done by examining which optimization methods for reducing the cost of computing a pairing can be applied to the case of using pairing-friendly curves of genus 2. The Kawazoe-Takahashi families of curves are an optimum choice for an effective pairing at the 192- and 256-bit security levels.

2. Detailed cost estimation of pairings on Kawazoe-Takahashi curves at the 192- and 256-bits security levels are provided in Chapter 5. Efficient constructions of the pairings is analyzed carefully, and one can see that the twisted Ate pairing is the best choice. For the twisted Ate pairing, the scaled cost is approximately 3.6 and 4.4 times more than the cost of pairing on the BLS12 curve at the 192-bit security level and on the BLS24 curve at the 256-bit security level, respectively.

   In addition, the benchmark results of the pairings on the Intel 64-bit platform are reported in Chapter 6. By using the library `mcl` which can generate optimum codes for field arithmetic and implementing multiplications in the first quadratic extension field in the lazy reduction way, the latency of our pairing is approximately 4 higher than the state-of-the-art in elliptic case for the 192-bit security level.

   This is the first comprehensive study of the detailed evaluations of pairings on genus 2 curves at the high security levels.

3. A new invariant of Weil-type pairing with an efficiently computable automorphism of a curve is proposed in Chapter 4. This result can be considered as a generalization of the methods that were used to construct the *omega pairing* on the elliptic curve by Zhao et al. [95]. Furthermore, this approach enables the Weil-type pairing in the Hess-Vercauteren framework. An effective method for computing the pairing which is evaluating the Miller function at effective parts of input divisors without normalizing the Miller function was provided.

2. Pairings on Hyperelliptic Curves of Genus 1 and 2

This chapter presents the previous works of pairings on genus 1 and 2 hyperelliptic curves. An overview of constructions of pairings and pairing-friendly hyperelliptic curves of genus 1 and 2 are given. Then the state-of-the-art results of pairings are described.

2.1 An Overview of Pairings

This section presents an overview of the various pairings on hyperelliptic curves. Let C be a hyperelliptic curve defined over $\mathbb{F}_q$ and let $\mathrm{Jac}_C(\simeq \mathrm{Pic}^0_C)$ denote the Jacobian variety of C. Let r be a positive integer and $\mathbb{F}_{q^k}$ be an extension field of $\mathbb{F}_q$ such that $r|(q^k-1)$ and $\mathrm{Jac}_C(\mathbb{F}_{q^k})$ contains no elements of the order r^2. The smallest integer k, which holds the above condition is known as the *embedding degree of Jac_C with respect to r*. For a divisor class $\overline{D} \in \mathrm{Jac}_C(\mathbb{F}_{q^k})[r]$, $f_{r,D}$ is known as the *Miller function* and denotes a rational function associated with the principal divisor rD. In general, for an integer i, the Miller function $f_{i,D}$ is defined as a rational function with divisor

$$(f_{i,D}) = iD - \rho(iD),$$

where $\rho(D)$ is the reduced divisor, which is equivalent to D. Note that we also denote $\rho(iD)$ by $[i]D$.

Let D and $E = \sum n_P P$ be divisors with disjoint support. Then the *modified Tate-Lichtenbaum pairing* t is defined as follows:

$$t\colon \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] \times \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] \to \mu_r \subset \mathbb{F}_{q^k}$$
$$(\overline{D}, \overline{E}) \mapsto f_{r,D}(E)^{(q^k-1)/r} = \left(\prod_P f_{r,D}(P)^{n_P}\right)^{(q^k-1)/r},$$

where μ_r is the group of r-th roots of unity.

The map t is bilinear and non-degenerate, and the value of t is independent of the representation of the divisor classes. After we evaluate the Miller function

$f_{r,D}$ at E, we need to perform the *final exponentiation* to obtain the pairing value $f_{r,D}(E)^{(q^k-1)/r}$.

One can also define the *Weil pairing*

$$\begin{aligned} e\colon \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] \times \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] &\to \mu_r \subset \mathbb{F}_{q^k} \\ (\overline{D}, \overline{E}) &\mapsto \frac{f_{r,D}(E)}{f_{r,E}(D)}. \end{aligned}$$

There is no final exponentiation when computing the Weil pairing.

By limiting the domains of pairings to the eigenspaces of the Frobenius map, more efficient pairings known as *Ate pairings* [44] and *twisted Ate pairings* [92], which have a shorter Miller loop, were exploited. These pairings are examples of *Hess-Vercauteren (HV) pairings* [7] constructed by Hess [49] and Vercauteren [90] as a general framework for pairings on Frobenius eigenspaces.

Let π be the q-th power Frobenius map. We take $\mathbb{G}_1$ and $\mathbb{G}_2$, which are subgroups of $\mathrm{Jac}_C(\mathbb{F}_{q^k})$, as follows,

$$\begin{aligned} \mathbb{G}_1 &:= \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] \cap \ker\,(\pi - [1]), \\ \mathbb{G}_2 &:= \mathrm{Jac}_C(\mathbb{F}_{q^k})[r] \cap \ker\,(\pi - [q]). \end{aligned}$$

Note that the group $\mathbb{G}_1$ is included in Jac_C over the defining field $\mathbb{F}_q$ and $\mathbb{G}_2$ is contained in $\mathrm{Jac}_C(\mathbb{F}_{q^k})\backslash\mathrm{Jac}_C(\mathbb{F}_q)$.

We consider $h(x) = \sum_{i=0}^{n} h_i x^i \in \mathbb{Z}[x]$ such that $h(s) \equiv 0 \pmod r$ for an integer s, and a *generalized Miller function* $f_{s,h,D}$, which is any function with

$$\sum_{i=0}^{n} h_i \rho(s^i D).$$

Let $s \equiv q^i \pmod r$ for some $j \in \mathbb{Z}$. We then obtain the *HV pairing* [7, Theorem 4.1]

$$\begin{aligned} a_{s,h}\colon \mathbb{G}_2 \times \mathbb{G}_1 &\to \mu_r \\ (\overline{D_2}, \overline{D_1}) &\mapsto f_{s,h,D_2}(D_1)^{(q^k-1)/r} \end{aligned}$$

satisfying

$$a_{s,h}(\overline{D_2}, \overline{D_1}) = t(\overline{D_2}, \overline{D_1})^{h(s)/r},$$

where D_1 is disjoint from D_2. $a_{s,h}$ is non-degenerate if and only if $h(s) \not\equiv 0 \pmod{r^2}$. Since it is possible to select a polynomial $h(x)$, where $|h_i| \leqslant r^{1/\varphi(k)}$ [90], the HV pairing is a product of Miller functions with loop length $(\log r)/\varphi(k)$, which is the optimal loop length.

If C has a twist C_t of degree d, i.e., d is the minimal integer satisfying that there exists an isomorphism $\phi\colon C_t \to C$ over $\mathbb{F}_{q^d}$, then a twisted version of the HV pairing and the Weil pairing in the HV framework exists [7, Remark 4.4],[49, Theorem 1]. In the twisted case, we remark that the automorphism $[\xi_m]\pi^e$ plays an important role, where $m = \gcd(k, d)$, $e = k/m$, and ξ_m is a unique primitive m-th root of unity. For more details, see [92, 50]. Then $[\xi_m]\pi^e$ acts on $\mathbb{G}_2$ as $[1]$ (i.e., the identity map) and $[\xi_m]$ acts on $\mathbb{G}_1$ as $[q^e]$ (i.e., $[\xi_m]\pi^e$ acts as $[q^e]$); therefore, we can reverse the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ in HV pairings.

For $s_e = p^{ei} \pmod{r}$, we then have that

$$\begin{aligned} a^{\text{twist}}_{s_e,h}\colon \mathbb{G}_1 \times \mathbb{G}_2 &\to \mu_r \\ (\overline{D_1}, \overline{D_2}) &\mapsto f_{s_e,h,D_1}(D_2)^{(q^k-1)/r}, \end{aligned}$$

and

$$\begin{aligned} e_{s_e,h}\colon \mathbb{G}_1 \times \mathbb{G}_2 &\to \mu_r \\ (\overline{D_1}, \overline{D_2}) &\mapsto \frac{f_{s_e,h,D_1}(D_2)}{f_{s_e,h,D_2}(D_1)} \end{aligned}$$

satisfying

$$a^{\text{twist}}_{s_e,h}(\overline{D_1}, \overline{D_2}) = t(\overline{D_1}, \overline{D_2})^{h(s)/r},\ e_{s_e,h}(\overline{D_1}, \overline{D_2}) = e(\overline{D_1}, \overline{D_2})^{h(s)/r}$$

are also bilinear and non-degenerate pairings [49, Theorem 1].

In general, the above twisted HV pairing $a^{\text{twist}}_{s_e,h}$ and the Weil-type HV pairing $e_{s_e,h}$ cannot be computed as a product of Miller functions with an optimal loop length for $e > 1$. Aranha et al. [2, 5] provided the β *Weil pairing*

$$\begin{aligned} \beta\colon \mathbb{G}_1 \times \mathbb{G}_2 &\to \mu_r \\ (\overline{D_1}, \overline{D_2}) &\mapsto \prod_{i=0}^{e} \frac{f_{q,h,[q^i]D_1}(D_2)}{f_{q,h,D_2}([q^i]D_1)}, \end{aligned}$$

which can be computed as $2e$ generalized Miller functions with an optimal loop length. In [2], the authors showed that the β pairings on various families of

pairing-friendly elliptic curves are well-suited for computing a single pairing in parallel on a multi-processor.

2.2 Pairing-Friendly Curves

Curves with somewhat small embedding degree and large prime-order subgroup of their Jacobian variety are suitable for constructing secure and efficient pairings. For a $\mathrm{Jac}_C(\mathbb{F}_{q^k})[r]$, the size of prime order r and the size of extension field $\mathbb{F}_{q^k}$ must be larger than corresponding key sizes determined by each security level. The detailed security levels and the corresponding bit size of keys are described in Section 3.1. $\rho = (g \log q)/\log r$ is known as the *ρ-value*, which is a characteristic to indicate their effectiveness for constructing a pairing. The definitions of *pairing-friendly* can be found in some literature. By [35, Definition 2.3], an elliptic curve $E(\mathbb{F}_{q^k})$ is pairing-friendly if the following condition hold:

1. there is a prime $r \geqslant \sqrt{q}$ dividing $\sharp E(\mathbb{F}_{q^k})$, and
2. $k < (\log_2 r)/8$.

And Balakrishnan et al. [7, Section 3.1] noted that a Jacobian variety is pairing-friendly if it holds that

$$k \leqslant 60 \text{ and } r > 2^{160}.$$

This section briefly reviewed pairing-friendly curves on elliptic and genus 2 hyperelliptic curves.

2.2.1 Pairing-Friendly Elliptic Curves

In the elliptic case, Freeman et al. [35] provided the comprehensive taxonomy of pairing-friendly curves including their original constructions. The essential method for constructing hyperelliptic curves is the complex multiplication (CM) method [85].

For constructions of supersingular curves, their cardinality can be computed easily. It is therefore easy to construct the curves. However, the supersingular curves with high embedding degree can be defined only the fields of small characteristics, there is no hope to obtain an effective pairing on these curves as noted in Section 3.1.

Cocks and Pinch offered an algorithm for constructing pairing-friendly (ordinary) curves with arbitrary embedding degree in their unpublished manuscript [27]. Note that the Cocks-Pinch method can be generalized for constructing hyperelliptic curves with higher genus [34, 36]. Brezing and Weng [25], and Baretto, Lynn and Scott [14] proposed the methods that construct parametric *families* of pairing-friendly curves where cardinalities of defining fields $q(x)$ and prime orders $r(x)$ are parametrized. In the elliptic case, we have the following suitable families of pairing-friendly curves: Kachisa-Schaefer-Scott (KSS) [55], Barreto-Naehrig (BN) [15], and Barreto-Lynn-Scott (BLS) [14]. They have $1 \leqslant \rho < 2$.

2.2.2 Pairing-Friendly Hyperelliptic Curves of Genus 2

In the genus 2 case, we do not have an *optimal* (i.e., the ρ-value is approximately 1) pairing-friendly approach yet. Freeman [34] proposed the construction of genus 2 curves of the prescribed embedding degree with a ρ-value around 8. Freeman and Satoh [37] constructed pairing-friendly genus 2 curves by using pairing-friendly elliptic curves over a finite extension of the base finite field. Kawazoe and Takahashi [58] provided an algorithm for constructing genus 2 curves of the form $y^2 = x^5 + ax$ by the closed formula for their Jacobian properties. An extended result for [58] is given by [54]. The result by Guillevic and Vergnaud [47] can be considered as a generalization of [37, 58]. The curves constructed by [37, 47, 54, 58] have ρ-values in the range $2 < \rho < 4$. Appropriate pairing-friendly genus 2 curves (cyclotomic families of curves obtained by the Brezing-Weng method) are presented in Table 1 in [47].

2.3 State-of-the-Art

This section reports the state-of-the-art of pairings on genus 1 and 2 curves with a focus on high security levels.

2.3.1 Pairings on Elliptic Curves

In the elliptic case, many pairings have been exploited, and many fast implementations of pairings at high security levels have been considered. For the 128-bit security level, an optimal Ate pairing on BN curves [15] is ideally suitable and

several implementation results for the pairings are proposed [17, 31, 56, 71]. As noted previously, Aranha et al. [2] showed the optimum choice of curves for a fast pairing at the 192-bit security level is the BLS family of embedding degree 12. Scott [81] offered many important results of pairing implementations on pairing-friendly families of curves. Scott used BLS24 curves for the 256-bit security level, and this is the best choice in this case.

It is also important to consider an effective method of fixed-argument pairings or products of pairings for a particular protocol [78, 81]. Zhang and Lin [93] analyzed which curves are suitable for computing products and quotients of pairings. According to their results, the best choice is KSS16 curves for the 192-bit security level, and BLS27 curves might be better for the 256-bit security level.

The table 1 shows the recommended choice of pairings on elliptic curves.

Table 1. Recommended pairings on elliptic curves at high security levels

Security level	Pairing
128	optimal Ate on BN12 curves
192	optimal Ate on BLS12 curves
256	optimal Ate on BLS24 curves

2.3.2 Pairings on Genus 2 Curves

In [26, 74], a supersingular genus 2 curve with embedding degree $k = 4$ is used for evaluation of the pairings. Their curves, however, are not suitable for high security levels.

The pairing on a genus 2 curve competitive with pairings of elliptic curves for the first time was the η_T pairing on the supersingular curves over binary fields [12]. Galbraith et al. [41] showed the explicit supersingular curve equations with embedding degree $k \in \{4, 5, 6, 12\}$. For $k = 6$, a given curve $C\colon y^2 = x^6 + 1$ is one of real model curves. Galbraith, Lin and Mireles Morales [39] provided an R-ate [64] pairing on C for the 80-bit security level.

Fan et al. [32] proposed a modified Tate pairing on a family of ordinary curves with embedding degree $k = 4$ with an efficiently computable automorphism. They evaluated the pairing for the 80-bit security level. In this research, a Weil-type

pairing with an efficiently computable automorphism is given in Section 4, and this pairing is more effective for high security levels.

A Twisted Ate pairing on genus 2 curves is defined by Zhang [92]. In the literature, the ordinary curves with twists of degree 8 are used to construct Ate and twisted Ate pairing without any detailed cost estimation.

3. Selection of Appropriate Genus 2 Curves for Efficient Pairings

This chapter considers how we should select a pairing-friendly genus 2 curve for constructing an efficient pairing. At first the target security level is clarified, and then several methods for speeding up pairings are explained. The requirements for constructing a fast pairing with an appropriate pairing-friendly curve are organized in order to facilitate the selection of appropriate curves.

3.1 Security Levels

For the r-torsion subgroup of Jacobian variety $\mathrm{Jac}_C(\mathbb{F}_q)[r]$ and the embedding degree k, the bit lengths of r and q^k should be at least 384 and 7936, respectively [18, NIST and ECRYPT II Recommendations], to construct a pairing at the 192-bit security level. For the 256-bit security level, the bit lengths of r and q^k should be at least 512 and 15424, respectively. The table 2 shows key sizes, that are size of subgroups r and size of extension fields q^k, and embedding degrees with various ρ-values for desired security levels. Note that genus g in the table should be 1 or 2.

Table 2. Recommended key sizes for pairings at desired security levels following [18, NIST and ECRYPT II Recommendations]

Security level	Subgroup size (r)	Extension field size (q^k)	Embedding degree (k)			
			$\rho \approx 1$	$\rho \approx 2$	$\rho \approx 3$	$\rho \approx 4$
128	256	3248	$12.7g$	$6.3g$	$4.2g$	$3.2g$
192	384	7936	$20.6g$	$10.3g$	$6.9g$	$5.2g$
256	512	15424	$30.1g$	$15.0g$	$10.0g$	$7.5g$

Recently, a major theoretical and practical breakthrough in the computation of discrete logarithms in finite fields of small characteristics and also other fields was made [8, 10]. As a result, type 1 [40] (symmetric) pairings defined on supersingular curves over finite fields of small characteristics have been vulnerable. Note that the most efficient pairing on genus 2 curves was an η_T pairing over the binary field [12]. Some type 1 pairings, however, still exist and are obtained

by using supersingular curves over large characteristic fields in the elliptic case [89, 94] and in the genus 2 case [39] that are unsuitable for achieving high security levels because of their small embedding degrees. We should therefore select *ordinary* pairing-friendly curves of genus 2 for our pairings.

Very recently, Kim and Barbulescu [59] provided a new variant of the Number Field Sieve algorithm which is known as exTNFS (extended tower number field sieve) for discrete logarithms in $\mathbb{F}_{p^n}$ and improved the asymptotic complexity. Their proposal method improved the polynomial selection by combining the TNFS (tower number field sieve) [11] construction with the conjugation method [9]. Furthermore, the methods sTNSF [53] and sexTNFS [51, 59] for the characteristic p which has a special form affect the key length of pairings on families of pairing-friendly curves. For the usual L_Q-notation

$$L_Q(\ell, c) = \exp\left((c + o(1))(\log Q)^{\ell}(\log\log Q)^{1-\ell}\right)$$

where $\mathbb{F}_Q = \mathbb{F}_{p^n}$, the current key sizes are chosen for the asymptotic complexity given by $L_Q(1/3, (64/9)^{1/3})$. For the case that p has a special form which is given by a polynomial, we should use $L_Q(1/3, (32/9)^{1/3})$ asymptotic complexity by the sexTNFS algorithm.

Hence we should extend the size of extension fields $\mathbb{F}_{p^k}$ for constructing a secure pairing in future. However, it remains too early to determine the bit sizes of the keys for the security levels with their complexity formulas. We therefore take the key sizes described as above for our pairings.

3.2 Speeding Up Pairings

Many techniques for speeding up computations of pairings have been exploited in the literature. For the Tate pairing $t(D_1, D_2) = f_{r,D_1}(D_2)^{(q^k-1)/r}$ where $D_1, D_2 \in \mathrm{Jac}(\mathbb{F}_{q^k})[r]$, the Miller function evaluated at D_2: $f_{r,D_1}(D_2)$ can be computed by the following standard Miller's algorithm.

For f computed by Miller's algorithm, we obtain the desired value by performing the final exponentiation $f^{(q^k-1)/r}$.

Algorithm 1 Miller's algorithm

INPUT: A Reduced divisor $D_1 \in \mathrm{Jac}_C(\mathbb{F}_{q^k})[r]$ and a divisor $D_2 \in \mathrm{Jac}_C(\mathbb{F}_{q^k})[r]$ where $D_1 \cap D_2 = \varnothing$, and an integer $s = \sum_{i=0}^{N} s_i 2^i$

OUTPUT: $f_{s,D_1}(D_2)$

1: $D \leftarrow D_1$
2: $f \leftarrow 1$
3: **for** $i = N - 1$ downto 0 **do**
4: Compute $[2]D$ and line function G where $[2]D = 2D - \mathrm{div}(G_{D,D})$
5: $D \leftarrow [2]D$
6: $f \leftarrow f^2 \cdot G_{D,D}(D_2)$
7: **if** $s_i = 1$ **then**
8: Compute $\rho(D + D_1)$ and line function G where $\rho(D + D_1) = D + D_1 - \mathrm{div}(G_{D,D_1})$
9: $D \leftarrow \rho(D + D_1)$
10: $f \leftarrow f \cdot G_{D,D_1}(D_2)$
11: **end if**
12: **end for**
13: **return** f

3.2.1 Field Constructions

Koblitz and Menezes [62] proposed *pairing-friendly fields* $\mathbb{F}_{p^k}$ (not to be confused with pairing-friendly curves) where $p = 1 \pmod{12}$ and k is of the form $2^i 3^j$. In this case, $\mathbb{F}_{p^k}$ can be constructed by an irreducible polynomial $x^k - \alpha$ where α is neither a square nor a cube in $\mathbb{F}_p$.

As noticed in [16, Section 4], the condition that $p = 1 \pmod{12}$ is not applicable to some cases of pairing-friendly elliptic curves. Instead, Benger and Scott proposed a *towering-friendly field* $\mathbb{F}_{q^m}$ [16, Definition 2] where q is a prime power, for which all prime divisors of m also divide $q - 1$. Then a towering-friendly field $\mathbb{F}_{q^m}$ is the tower of sub-extensions which can be constructed by binomials. We can therefore implement a pairing effectively in a towering-friendly field.

3.2.2 Squaring in Cyclotomic Subgroups

In the literature, cyclotomic subgroups of the multiplicative group of finite fields have been used for compressing representation of elements in the cyclotomic subgroup for public-key cryptosystems [82, 86, 87]. Granger and Scott [46] extended the methods to perform arithmetic, especially a squaring in the extension field using the cyclotomic subgroups offered by [88, 45]. Indeed, they showed that a squaring in the order $\Phi_6(q)$ cyclotomic subgroup of $\mathbb{F}_{q^6}$

$$G_{\Phi_6(q)} = \{a \in \mathbb{F}_{q^6} \mid a^{\Phi_6(q)} = a^{q^2-q+1} = 1\}$$

requires 3 squarings in $\mathbb{F}_{q^2}$ in $q = 1 \pmod 6$.

Karabina [57] provided a new squaring formulae and comprehensive squaring formulas in $G_{\Phi_6(q)}$ including the results by Granger and Scott by computing Gröbner bases. These squaring formulas are effective for performing a final exponentiation since $g^{\Phi_k(q)} = 1$ for $g = f^{\frac{q^k-1}{\Phi_k(q)}}$ where

$$f^{\frac{q^k-1}{r}} = f^{\frac{q^k-1}{\Phi_k(q)} \cdot \frac{\Phi_k(q)}{r}}. \tag{1}$$

If $q = 1 \pmod 6$ and k is divisible by 6, we can apply the squaring methods to compute the exponentiation $g^{\frac{\Phi_k(q)}{r}}$ (the hard part of the final exponentiation).

3.2.3 Final Exponentiation

As noted in the previous subsection, the final exponentiation (1) can be divided into two parts, i.e., computing $g = f^{\frac{q^k-1}{\Phi_k(q)}}$ and computing $g^{\frac{\Phi_k(q)}{r}}$ which is known as the *hard part*. It is easy to compute g since this computation can be performed with p^i-Frobenius effectively.

For simplicity, we suppose that $q = p$. For given parameters r, p, and k, the dedicated methods for performing the hard parts has been exploited [28, 48]. Scott et al. [83] presented a procedure to speed up a hard part with computing vectorial addition chain for performing a multi-exponentiation. Especially, their method is useful for families of curves, that is, the case that parameters $r(x)$ and $p(x)$ is parametrized by a variable x [4, 17]. Their method can be presented briefly as follows. For $l = \Phi_k(p)/r$, we write l as a p-adic number

$$l = l_0 + l_1 p + \cdots + l_{\varphi(k)-1} p^{\varphi(k)-1}.$$

After performing p^i-Frobenius g^{p^i} we then compute g^l by performing a multi-exponentiation with a short vectorial addition chain derived from the coefficients l_i. Indeed, the addition chain can be computed using the set of coefficients for the polynomials $l_i(x)$ where $l_i(x)$ can also be parametrized with $r(x)$ and $p(x)$. On the whole, the computation of hard part with the method by Scott et al. consists of three part:

1. computing g^{p^i},
2. performing h^x several times for computing g^{x^i} where h is an element in the cyclotomic subgroup,
3. performing a multi-exponentiation with a short vectorial addition chain in order to obtain g^l.

3.2.4 Arithmetic in Divisor Class Group

This subsection presents Cantor's algorithm for genus 2 curves for adding general divisor classes. In contrast to the elliptic case, arithmetic in Jacobian group is more complicated. We should therefore reduce the cost as much as possible. Algorithm 2 presents the standard Cantor's algorithm in Jac_C for

$$C\colon y^2 + H(x)y = F(x).$$

Note that we can obtain the rational function $(d(x)(y - v(x)))/u'(x)$ where

$$\rho(D_1 + D_2) = D_1 + D_2 - \mathrm{div}\left(\frac{d(x)(y - v(x))}{u'(x)}\right). \tag{2}$$

By using the notations in Algorithm 1 we have $G_{D_1,D_2} = (d(x)(y - v(x)))/u'(x)$.

In general case, we can perform a doubling in $\mathrm{Jac}_C(\mathbb{F}_p)$ with 22 multiplications, 5 squarings and an inversion in $\mathbb{F}_p$, and an addition with 22 multiplications, 3 squarings and an inversion in $\mathbb{F}_p$ using affine coordinate system [63]. In the projective coordinate system, a doubling requires 38 multiplications and 6 squarings, and an addition requires 47 multiplications and 4 squarings in $\mathbb{F}_p$ respectively.

Fan, Gong, and Jao provided the explicit formula for adding divisors and the dedicated coordinate system by [33] for

$$C\colon y^2 = f(x),\ f(x) = x^5 + f_3x^3 + f_2x^3 + f_1x + f_0$$

Algorithm 2 Cantor's algorithm [38, Algorithm 1]

INPUT: A Reduced divisor $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$.

OUTPUT: A reduced divisor $\rho(D_1 + D_2)$.

1: $d_1 \leftarrow \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2: $d \leftarrow \gcd(d_1, v_1 + v_2 + H) = c_1 d_1 + c_2(v_1 + v_2 + H)$
3: $s_1 \leftarrow c_1 e_1,\ s_2 \leftarrow c_1 e_2,\ s_3 \leftarrow c_2$
4: $u \leftarrow (u_1 u_2)/d^2,\ v \leftarrow (s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + F))/d \mod u$
5: **while** $\deg(u) > g$ **do**
6: $\quad u' \leftarrow \mathrm{Monic}((F - vH - v^2)/u),\ v' \leftarrow (-H - v) \mod u'$
7: $\quad u \leftarrow u',\ v \leftarrow v'$
8: **end while**
9: **return** $\rho(D_1 + D_2) = [u', v']$.

By using their method a doubling can be performed with 35 multiplications and 5 squarings in $\mathbb{F}_p$, and a mixed addition requires 36 multiplications and 5 squarings in $\mathbb{F}_p$.

3.2.5 Denominator Elimination

In the Miller loop (Algorithm 1), the rational functions $G_{D,D}$ and G_{D,D_1} are evaluated the input divisor D_2. We suppose that $G_{D,D}$ or G_{D,D_1} is of the form $(y - v(x))/u'(x)$ as described in the previous subsection. Assume that D_1 is defined over $\mathbb{F}_p$. If $D_2 = [u_2(x), v_2(x)]$ has $u_2(x)$ defined over proper subfield $\mathbb{F}_{p^d}$ of $\mathbb{F}_{p^k}$, the denominator of $G_{D,D}$ or G_{D,D_1} evaluated at D_2 is defined over $\mathbb{F}_{p^d}$, therefore the value becomes trivial after final exponentiation. This optimization is known as *denominator elimination* [13].

3.3 Selection of Curves

A construction of an efficient pairing requires us to use a pairing-friendly curve C with the following features:

- C has a higher degree twist such that we can reduce the costs of arithmetic in the extension fields.

- The extension field $\mathbb{F}_{q^k}$ can be constructed as a tower field such as [16].
- The form in which a polynomial for C is defined is simple such that we can use an explicit formula for computing divisors in Jac_C.

We first consider the twists of C. In the genus 2 case, we can obtain a twist of degree at most 10, where $\mathrm{char}(q) > 5$. As described in [92, Section 3], for the curve

$$C\colon y^2 = x^5 + a$$

over $\mathbb{F}_q$, where $q = 1 \pmod 5$, we have a twist of degree 10

$$C_t\colon y^2 = x^5 + ac \ (c \in \mathbb{F}_q),$$

where

$$\psi\colon \mathrm{Jac}_{C_t} \to \mathrm{Jac}_C;\ (x, y) \mapsto (c^{-\frac{1}{5}}x, c^{-\frac{1}{2}}y).$$

The form of ψ does not permit the use of the denominator elimination method to compute the pairing. Furthermore, we do not have an explicit method to generate curves of such form with a lower ρ-value. Hence, it is necessary to consider other curves.

Then, it is natural to consider a curve with degree 8. The methods by Kawazoe and Takahashi [58] provide families of curves

$$C\colon y^2 = x^5 + ax,$$

and these curves have a twist

$$C_t\colon y^2 = x^5 + acx \ (c \in \mathbb{F}_q)$$

if $q = 1 \pmod 8$ and we have that

$$\psi\colon \mathrm{Jac}_{C_t} \to \mathrm{Jac}_C;\ (x, y) \mapsto (c^{-\frac{1}{4}}x, c^{-\frac{1}{8}}y).$$

In this case we can use the denominator elimination method. In addition, we can perform arithmetic in the divisor class group by using an explicit formula and a dedicated coordinate system [33].

For efficient arithmetic in the extension fields, the embedding degree k should be of the form $2^i 3^j$. In accordance with the security parameters q, r, we should take $k = 16$ and $k = 24$ for the 192-bit and 256-bit security levels, respectively. As described in Section 5.4.4, we can use the methods of squaring in the cyclotomic subgroup $G_{\Phi_6(p^4)}$, where $k = 24$, by specifying appropriate parameters for the curve.

We conclude that Kawazoe-Takahashi curves are the most efficient for constructing a pairing at the high security level.

3.4 Summary

This chapter proposed that the Kawazoe-Takahashi cyclotomic families are suitable for constructing pairings for the high security levels in genus 2 case. The selection of curves is based on the analysis of several methods for speeding up computations of pairings, e.g., efficient constructions of extension fields, suitable twists for adopting the denominator elimination technique, and so on.

4. The Optimal Weil Pairings with the Efficiently Computable Automorphisms

This chapter describes some Weil-type pairings on the Kawazoe-Takahashi curves C with their automorphisms.

4.1 Main Results

Instead of the β pairing described in Section 2.1 and the Weil-type pairing in the Hess-Vercauteren framework, we evaluate Weil pairings with automorphism given as follows.

Theorem 1. *Let m be an integer satisfying $\lambda^4 + 1 = mr$, and $\beta = (4\lambda^3)^{-1} \pmod r$. For $D_1 \in \mathrm{Jac}_C(\mathbb{F}_p)[r]$ and $D_2' \in \mathrm{Jac}_{C_t}(\mathbb{F}_p)[r]$ where $D_1 \cap \psi(D_2') = \varnothing$, suppose that $\varphi(D_1) = [\lambda]D_1$, $\hat{\varphi}'(D_2') = [\lambda]D_2'$ and D_2' is reduced. Then*

$$\begin{aligned}\left(\frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,D_2'}(\psi^{-1}(D_1))}\right)^{p^{k/2}-1} &= \left(\frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,\psi(D_2')}(D_1)}\right)^{p^{k/2}-1} \\ &= e(\overline{D_1}, \overline{\psi(D_2')})^{\beta m(p^{k/2}-1)}\end{aligned}$$

is a non-degenerate bilinear pairing.

This result can be considered as a generalization of the methods that were used to construct the *omega pairing* on the elliptic curve by Zhao et al. [95]. This approach enables the Weil-type pairing in the Hess-Vercauteren framework to be computed as follows.

Theorem 2. *For the divisors D_1, D_2' in Theorem 1 and $h(x) \sum_{i=0}^{n} h_i x^i \in \mathbb{Z}[x]$ where $h(\lambda) \equiv 0 \pmod r$,*

$$\begin{aligned}e_{\lambda,h}(\overline{D_1}, \overline{\psi(D_2')})^{p^{k/2}-1} &= \left(\frac{f_{\lambda,h,D_1}(\psi(D_2'))}{f_{\lambda,h,\psi(D_2')}(D_1)}\right)^{p^{k/2}-1} \\ &= \left(\frac{f_{\lambda,h,D_1}(\psi(D_2'))}{f_{\lambda,h,D_2'}(\psi^{-1}(D_1))}\right)^{p^{k/2}-1}\end{aligned}$$

is a non-degenerate bilinear pairing satisfying

$$e_{\lambda,h}(\overline{D_1}, \overline{\psi(D_2')})^{p^{k/2}-1}$$
$$= e(\overline{D_1}, \overline{\psi(D_2')})^{m(p^{k/2}-1)} \cdot \left(\frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,\psi(D_2')}(D_1)} \right)^{\left(\sum_{i=1}^{\deg h} -h_i i\lambda^{i-1}\right)\left(p^{k/2}-1\right)}.$$

In order to prove Theorem 1 and 2, we obtain the following results.

Lemma 4.1. *In a situation similar to above, we have that*

$$\left(\frac{f_{\lambda,D_1}([\lambda^i]\psi(D_2'))}{f_{\lambda,D_2'}([\lambda^i]\psi^{-1}(D_1))} \right) = \left(\frac{f_{\lambda,D_1}([\lambda^{i-1}]\psi(D_2'))}{f_{\lambda,D_2'}([\lambda^{i-1}]\psi^{-1}(D_1))} \right)^{\lambda}.$$

Proof. It suffices to show

$$f_{\lambda,D_1}([\lambda^i]\psi(D_2'))f_{\lambda,D_1}([\lambda^{i-1}]\psi(D_2'))^{-\lambda}$$
$$= f_{\lambda,D_2'}([\lambda^i]\psi^{-1}(D_1))f_{\lambda,D_2'}([\lambda^{i-1}]\psi^{-1}(D_1))^{-\lambda}.$$

We then have

$$\begin{aligned}
&f_{\lambda,D_1}([\lambda^i]\psi(D_2'))f_{\lambda,D_1}([\lambda^{i-1}]\psi(D_2'))^{-\lambda} \\
&= f_{\lambda,D_1}([\lambda^i]\psi(D_2') - \lambda[\lambda^{i-1}]\psi(D_2')) \\
&= f_{\lambda,D_1}(-\mathrm{div}(f_{\lambda,[\lambda^{i-1}]\psi(D_2')})) \\
&= f_{\lambda,[\lambda^{i-1}]\psi(D_2')}([\lambda]D_1 - \lambda D_1) \\
&= f_{\lambda,\hat{\varphi}^{i-1}(\psi(D_2'))}([\lambda]D_1 - \lambda D_1) \\
&= f_{\lambda,\psi(D_2')}(\varphi^{i-1}([\lambda]D_1 - \lambda D_1)) \\
&= f_{\lambda,\psi(D_2')}([\lambda^i]D_1 - \lambda[\lambda^{i-1}]D_1)) \\
&= f_{\lambda,D_2'}(\psi^{-1}([\lambda^i]D_1 - \lambda[\lambda^{i-1}]D_1)) \\
&= f_{\lambda,D_2'}([\lambda^i]\psi^{-1}(D_1) - \lambda[\lambda^{i-1}]\psi^{-1}(D_1)).
\end{aligned}$$

□

For a rational function f with order $-r$, the *leading coefficient at* P_∞ denoted by $\mathrm{lc}_\infty(f)$ can be defined by $(z^r f)(P_\infty)$ where z is a uniformizer at P_∞.

Lemma 4.2. *For $D_2' \in C_t(\mathbb{F}_{p^e})$, we have*

$$f_{i,\psi(D_2')} = \alpha \cdot (f_{i,D_2'} \circ \psi^{-1})$$

since $\mathrm{div}(f_{i,\psi(D_2')}) = \mathrm{div}(f_{i,D_2'} \circ \psi^{-1})$. *Suppose that D_2' is reduced. Then*

$$\alpha = \begin{cases} c^{-\frac{2(i-1)}{d}} \cdot \dfrac{\mathrm{lc}_\infty\left(f_{i,\psi(D_2')}\right)}{\mathrm{lc}_\infty\left(f_{i,D_2'}\right)} & (r \nmid i), \\ c^{-\frac{2i}{d}} \cdot \dfrac{\mathrm{lc}_\infty\left(f_{i,\psi(D_2')}\right)}{\mathrm{lc}_\infty\left(f_{i,D_2'}\right)} & (r \mid i). \end{cases}$$

Proof. First we consider the case $r \nmid i$. Let $z = x^2/y$ be a uniformizer at the infinity point P_∞. The Miller functions $f_{i,D_2'}$ and $f_{i,\psi(D_2')}$ have a pole of order $2(i-1)$ at P_∞. It therefore holds that

$$\begin{aligned} \frac{f_{i,\psi(D_2')}}{f_{i,D_2'} \circ \psi^{-1}} &= \frac{\frac{f_2}{z^{2(i-1)}}}{\frac{f_1}{z^{2(i-1)}} \circ \psi^{-1}} \\ &= \frac{z^{2(i-1)} \circ \psi^{-1}}{z^{2(i-1)}} \cdot \frac{f_2}{f_1 \circ \psi^{-1}} \\ &= c^{-\frac{2(i-1)}{d}} \cdot \frac{f_2}{f_1 \circ \psi^{-1}} \end{aligned}$$

where we write that

$$f_{i,\psi(D_2')} = \frac{f_2}{z^{2(i-1)}}, \ f_{i,D_2'} = \frac{f_1}{z^{2(i-1)}}$$

with the uniformizer $z = x^2/y$. We can evaluate the function at P_∞ and obtain

$$\alpha = c^{-\frac{2(i-1)}{d}} \cdot \frac{\mathrm{lc}_\infty(f_{i,\psi(D_2')})}{\mathrm{lc}_\infty(f_{i,D_2'})}.$$

When $r \mid i$, the proof is similar, where $f_{i,D_2'}$ and $f_{i,\psi(D_2')}$ have a pole of order $2i$ at P_∞. □

Proof of Theorem 1. We consider that

$$\frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,\psi(D_2')}(D_1)} = \frac{1}{\alpha} \cdot \frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,D_2'}(\psi^{-1}(D_1))}.$$

By using [12, Lemma 2,3] (the divisors D_1 and $\psi(D_2')$ do not need to be reduced), we have that

$$\begin{aligned} f_{mr,D_1} &= f_{\lambda^4+1,D_1} \\ &= \gamma_1 \cdot f_{\lambda^4,D_1} \cdot u(x) \\ &= \gamma_1 \cdot \left(\prod_{i=0}^{3} f_{\lambda,[\lambda^i]D_1}^{\lambda^{3-i}} \right) \cdot u_1(x) \end{aligned}$$

where $\gamma_1 \in \mathbb{F}_p$ and $D_1 = [u_1(x), v_1(x)]$ with the Mumford representation. Similarly, it holds that

$$f_{mr,D_2'} = \gamma_2 \cdot \left(\prod_{i=0}^{3} f_{\lambda,[\lambda^i]D_2'}^{\lambda^{3-i}} \right) \cdot u_2'(x),$$

where $\gamma_2 \in \mathbb{F}_{p^e}$ and $D_2' = [u_2'(x), v_2'(x)]$ with the Mumford representation. From Lemma 4.1, we have that

$$\begin{aligned} \frac{f_{\lambda,[\lambda^i]D_1}^{\lambda^{3-i}}(\psi(D_2'))}{f_{\lambda,[\lambda^i]D_2'}^{\lambda^{3-i}}(\psi^{-1}(D_1))} &= \theta_i \cdot \left(\frac{f_{\lambda,D_1}([\lambda^i]\psi(D_2'))}{f_{\lambda,D_2'}([\lambda^i]\psi^{-1}(D_1))} \right)^{\lambda^{3-i}} \\ &= \theta_i \cdot \left(\frac{f_{\lambda,D_1}([\lambda^i]\psi(D_2'))}{f_{\lambda,D_2'}([\lambda^i]\psi^{-1}(D_1))} \right)^{\lambda^3} \end{aligned}$$

where $\theta_i \in \mathbb{F}_p$. We therefore obtain that

$$\begin{aligned} e(D_1, \psi(D_2'))^{m(p^{k/2}-1)} &= \left(\frac{\gamma_1\theta}{\gamma_2\alpha^m} \cdot \left(\frac{f_{\lambda,D_1}([\lambda^i]\psi(D_2'))}{f_{\lambda,\psi(D_2')}([\lambda^i]D_1)} \right)^{4\lambda^3} \cdot \frac{u_1(\psi(D_2'))}{u_2'(\psi^{-1}(D_1))} \right)^{p^{k/2}-1} \\ &= \left(\frac{f_{\lambda,D_1}([\lambda^i]\psi(D_2'))}{f_{\lambda,\psi(D_2')}([\lambda^i]D_1)} \right)^{4\lambda^3(p^{k/2}-1)}, \end{aligned}$$

where $\theta = \prod_{i=0}^{3} \theta_i$ since $\frac{u_1(\psi(D_2'))}{u_2'(\psi^{-1}(D_1))} \in \mathbb{F}_{p^{k/2}}$ and α is contained in a proper subfield of $\mathbb{F}_{p^k}$ when the Miller functions are normalized by Lemma 4.2. □

Proof of Theorem 2. Let m be an integer such that $h(\lambda) = mr$. For a divisor and the generalized Miller function $f_{s,h,D}$, it holds that

$$f_{s,h,D} = \prod_{i=0}^{\deg h} f_{h_i,[s^i]D} \cdot \prod_{i=1}^{\deg h} g_{[t_{i-1}]D,[h_i s^i]D},$$

where $t_i = \sum_{j=0}^{i} h_j s^j$ and $g_{[a]D,[b]D}$ is the normalized rational function with the divisor $[a]D + [b]D - [a+b]D$. By using this, we have that

$$\begin{aligned} f_{r,D_1}^m &= f_{\sum h_i \lambda^i, D_1} \\ &= \prod_{i=0}^{\deg h} f_{h_i\lambda^i,D_1} \cdot \prod_{i=1}^{\deg h} g_{[t_{i-1}]D,[h_i s^i]D} \\ &= \prod_{i=0}^{\deg h} f_{\lambda^i,D_1}^{h_i} \cdot \prod_{i=0}^{\deg h} f_{h_i,[\lambda^i]D_1} \cdot \prod_{i=1}^{\deg h} g_{[t_{i-1}]D,[h_i s^i]D} \\ &= \prod_{i=0}^{\deg h} f_{\lambda^i,D_1}^{h_i} \cdot f_{\lambda,h,D_1}. \end{aligned}$$

Hence, it holds that

$$f_{\lambda,h,D_1} = f_{r,D_1}^m \cdot \prod_{i=0}^{\deg h} f_{\lambda^i,D_1}^{-h_i}.$$

We have a similar result for $f_{\lambda,h,\psi(D_2)}$. We then obtain

$$\begin{aligned} \frac{f_{\lambda,h,D_1}}{f_{\lambda,h,\psi(D_2)}} &= e(D_1,\psi(D_2'))^m \cdot \prod_{i=0}^{\deg h} \left(\frac{f_{\lambda^i,D_1}}{f_{\lambda^i,\psi(D_2')}}\right)^{-h_i} \\ &= e(D_1,\psi(D_2'))^m \cdot \left(\frac{f_{\lambda,D_1}}{f_{\lambda,\psi(D_2')}}\right)^{\sum_{i=0}^{\deg h} -h_i i\lambda^{i-1}}. \end{aligned}$$

By Theorem 1,

$$\left(\frac{f_{\lambda,D_1}(\psi(D_2'))}{f_{\lambda,\psi(D_2')}(D_1)}\right)^{p^{k/2}-1}$$

is a pairing. Hence, we obtain the desired result. □

An efficient computation of Weil-type pairings requires us to select a reduced divisor as the first argument and evaluate the Miller function at an effective part of the reduced divisor such as the Weil pairing on points in the elliptic case [67, Proposition 8].

Lemma 4.3. *Suppose that D_1 and $\psi(D_2')$ are reduced, and $D_1 \cap \epsilon(\psi(D_2')) = \varnothing$. Then we have*

$$\left(\frac{f_{\lambda,\overline{D_1}}(\overline{\psi(D_2')})}{f_{\lambda,\overline{D_2'}}(\overline{\psi^{-1}(D_1)})}\right)^{2(p^{k/2}-1)} = \left(\frac{f_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1)))}\right)^{2(p^{k/2}-1)}.$$

Proof. Although we only consider the case $\lambda \nmid r$, the proof is almost the same when $\lambda \mid r$.

Let $\widetilde{D_1} = D_1 + \mathrm{div}(z^{b_1})$ be a divisor equivalent to D_1 and let $z = x^2/y$ be a uniformizer at P_∞. We assume that $\epsilon(D_2') \cap \mathrm{div}(z^{b_1}) = \varnothing$. If $O = (0,0) \in \mathrm{supp}(D_2')$ then we can take $(x-1)^2/y$ as z so that $\epsilon(D_2')$ and $\mathrm{div}(z^{b_1})$ have disjoint support. This does not affect the validity of the proof. We write $D_i = \epsilon(D_i) - b_i P_\infty$, where $b_i = \deg(\epsilon(D_i))$. Then

$$\begin{aligned}\lambda\widetilde{D_1} - [\lambda]\widetilde{D_1} &= \lambda D_1 + \mathrm{div}(z^{b_1\lambda}) - [\lambda]D_1 \\ &= \mathrm{div}(f_{\lambda,D_1} \cdot z^{b_1\lambda}),\end{aligned}$$

and we can see that

$$\begin{aligned}\frac{f_{\lambda,\overline{D_1}}(\overline{\psi(D_2')})}{f_{\lambda,\overline{D_2'}}(\overline{\psi^{-1}(D_1)})} &= \frac{f_{\lambda,\widetilde{D_1}}(\psi(D_2'))}{f_{\lambda,D_2'}(\psi^{-1}(\widetilde{D_1}))} \\ &= \frac{f_{\lambda,D_1} \cdot z^{b_1\lambda}(D_2')}{f_{\lambda,D_2'}(\psi^{-1}(\widetilde{D_1}))} \\ &= \frac{f_{\lambda,D_1}(\epsilon(D_2')) \cdot z^{b_1\lambda}(\epsilon(D_2'))}{f_{\lambda,D_2'}(\psi^{-1}(\widetilde{D_1})) \cdot (f_{\lambda,D_1} \cdot z^{b_1\lambda})(b_2 P_\infty)}.\end{aligned}$$

Now we have $z^{b_1\lambda} = x^{b_1^2\lambda}/y^{b_1\lambda}$ and $(c^{-\frac{5}{d}})^2 \in \mathbb{F}_{p^{k/2}}$. Hence, $z^{b_1\lambda}(\epsilon(D_2'))^{2(p^{k/2}-1)} = 1$. Moreover, it follows that

$$f_{\lambda,D_1} \cdot z^{b_1\lambda}(b_2 P_\infty)^{2(p^{k/2}-1)} = 1$$

since $f_{\lambda,D_1} \cdot z^{b_1\lambda}$ is $\mathbb{F}_p$-rational.

For the Miller function

$$f_{\lambda,D_2'}(\psi^{-1}(\widetilde{D_1})) = f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1))) \cdot f_{\lambda,D_2'}(\psi^{-1}(\mathrm{div}(z^{b_1}) - b_1 P_\infty)),$$

it suffices to show that

$$f_{\lambda,D_2'}(\psi^{-1}(\mathrm{div}(z^{b_1}) - b_1 P_\infty))^{2(p^{k/2}-1)} = 1.$$

Let $Q = (x_Q, y_Q) \in C_t(\mathbb{F}_p)$ be a general point satisfying that $\mathrm{div}(x - x_Q) \cap \mathrm{div}(f_{\lambda,D_2'}) = \varnothing$ where $\mathrm{div}(x - x_Q) = Q + \iota(Q) - 2P_\infty$. We write $P = (x_P, y_P) =$

$\psi(Q)$. Then the support of $\mathrm{div}(f_{\lambda,D_2'})$ and that of $\psi^{-1}(\mathrm{div}(x-x_P)+b_1P_\infty)$ are disjoint since

$$\psi^{-1}(\mathrm{div}(x-x_P)+b_1P_\infty) = Q+\iota(Q),$$

where ι is the hyperelliptic involution of C_t. Hence, we have

$$f_{\lambda,D_2'}(\psi^{-1}(\mathrm{div}(x-x_P)+b_1P_\infty))^{2(p^{k/2}-1)} = 1$$

by Lemma 1 of [30]. Therefore, it follows that

$$\begin{aligned}
&f_{\lambda,D_2'}(\psi^{-1}(\mathrm{div}(z^{b_1})-b_1P_\infty))^{2(p^{k/2}-1)}\\
&= \left(f_{\lambda,D_2'}(\mathrm{div}(z^{b_1})-b_1P_\infty)\cdot f_{\lambda,D_2'}(\psi^{-1}(\mathrm{div}(x-x_P)+b_1P_\infty)\right)^{2(p^{k/2}-1)}\\
&= \left(f_{\lambda,\psi(D_2')}(\mathrm{div}(z^{b_1}\cdot(x-x_P)))\right)^{2(p^{k/2}-1)}.
\end{aligned}$$

Since $\mathrm{div}(f_{\lambda,\psi(D_2')})\cap\mathrm{div}(z^{b_1})=\varnothing$, by using Weil reciprocity we obtain

$$\begin{aligned}
&\left(f_{\lambda,\psi(D_2')}(\mathrm{div}(z^{b_1}\cdot(x-x_P)))\right)^{2(p^{k/2}-1)}\\
&= \left((z^{b_1}\cdot(x-x_P))(\lambda\psi(D_2')-[\lambda]\psi(D_2'))\right)^{2(p^{k/2}-1)}.
\end{aligned}$$

The values evaluated by rational functions z^{2b_1} and $(x-x_P)$ at the divisors $\psi(D')$ ($D'\in\mathrm{Jac}_{C_t}(\mathbb{F}_{p^e})$) are in $\mathbb{F}_{p^{k/2}}$. Therefore, it holds that

$$\left((z^{b_1}\cdot(x-x_P))(\lambda\psi(D_2')-[\lambda]\psi(D_2'))\right)^{p^{k/2}-1} = 1.$$

□

Note that when D_1 is non-degenerate we have that

$$\left(\frac{f_{\lambda,\overline{D_1}}(\overline{\psi(D_2')})}{f_{\lambda,\overline{D_2'}}(\overline{\psi^{-1}(D_1)})}\right)^{2(p^{k/2}-1)} = \left(\frac{f_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1)))}\right)^{2(p^{k/2}-1)}.$$

since $b_1=2$ in the proof of the lemma.

The following result ensures that we do not need to normalize the Miller functions by raising the pairing value to the $(2(p^{k/2}-1))$-th power.

Lemma 4.4. *It holds that*

$$\mathrm{lc}_\infty\left(f_{i,\psi(D_2')}\right)^{2(p^{k/2}-1)} = 1.$$

Proof. In the Cantor algorithm, the leading coefficients of the line functions $\frac{y-v(x)}{u(x)}$, where $v(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ in the Miller loop, are given by

$$\mathrm{lc}_\infty\left(\frac{y-v(x)}{u(x)}\right) = -b_3.$$

When we compute

$$\psi(D) + \psi(D') = \psi(D'') + \mathrm{div}\left(\frac{y-v(x)}{u(x)}\right)$$

at a certain Miller step when computing $f_{i,\psi(D_2')}$, we write the corresponding polynomial as

$$\begin{aligned} v'(x) &= b_3'x^3 + b_2'x^2 + b_1'x + b_0', \\ D + D' &= D'' + \mathrm{div}\left(\frac{y-v'(x)}{u'(x)}\right). \end{aligned}$$

Then, we can see that

$$v(x) = c^{\frac{1}{d}}b_3x^3 + c^{-\frac{1}{d}}b_2x^2 + c^{-\frac{3}{d}}b_1x + c^{-\frac{5}{d}}b_0.$$

Indeed, for a point $\psi((x', y')) = (c^{-\frac{2}{d}}x', c^{-\frac{5}{d}}y')$, it holds that

$$v(c^{-\frac{2}{d}}x') = c^{-\frac{5}{d}} \cdot v(x') = c^{-\frac{5}{d}}y';$$

Therefore, all the points in $\mathrm{supp}(\psi(D) + \psi(D'))$ are on the curve defined by $y - v(x)$. Since $\mathrm{lc}_\infty\left(f_{i,\psi(D_2')}\right)$ is the product of $-c^{\frac{1}{d}}b_3$, $(b_3 \in \mathbb{F}_{p^e})$, it suffices to know that $(-c^{\frac{1}{d}}b_3)^{2(p^{k/2}-1)} = 1$. This holds by considering that $-c^{\frac{2}{d}} \in \mathbb{F}_{p^{k/2}}$. □

This result enables us to compute the Weil-type pairing with normalized Miller functions as

$$\left(\frac{f^{\mathrm{norm}}_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f^{\mathrm{norm}}_{\lambda,\psi(D_2')}(\epsilon(D_1))}\right)^{2(p^{k/2}-1)} = \left(\frac{f_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1)))}\right)^{2(p^{k/2}-1)}.$$

Note that when D_2' is non-degenerate, it requires us to raise the value to the $(p^{k/2}-1)$-th power since $\deg(\epsilon(D_2')) = 2$.

By using Lemma 4.3 and 4.4, we obtain an efficient method for computing the Weil pairing with automorphism.

Theorem 3. *Let m be an integer satisfying $\lambda^4 + 1 = mr$, and $\beta = (4\lambda^3)^{-1} \pmod r$. For reduced divisors $D_1 \in \mathrm{Jac}_C(\mathbb{F}_p)[r]$ and $D_2' \in \mathrm{Jac}_{C_t}(\mathbb{F}_p)[r]$, where $D_1 \cap \psi(D_2') = \varnothing$, suppose that $\varphi(D_1) = [\lambda]D_1$, $\hat{\varphi}'(D_2') = [\lambda]D_2'$. Then*

$$\left(\frac{f_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1)))}\right)^{2(p^{k/2}-1)} = \left(\frac{f^{\mathrm{norm}}_{\lambda,D_1}(\epsilon(\psi(D_2')))}{f^{\mathrm{norm}}_{\lambda,\psi(D_2')}(\epsilon(D_1))}\right)^{2(p^{k/2}-1)} = e(\overline{D_1}, \overline{\psi(D_2')})^{2\beta m(p^{k/2}-1)} \tag{3}$$

is a pairing. □

We omit the proof of the theorem.

4.2 Discussion

Fan et al. [32] provided the Tate pairing on the two families of non-supersingular genus 2 hyperelliptic curves with the efficiently computable automorphisms. Indeed, they used the Kawazoe-Takahashi curve

$$C\colon y^2 = x^5 + ax,\ a \in \mathbb{F}_p,\ p \equiv 1 \pmod 8$$

with embedding degree $k = 4$ for the 80-bit security level, and used the automorphism $\varphi\colon C \to C;\ (x, y) \mapsto (\zeta_8^2 x, \zeta_8 y)$ where $\zeta_8 \in \mathbb{F}_p$ is a primitive 8-th root of unity. The characteristic polynomial of φ is $T^4 + 1$ and let λ be a root of the characteristic polynomial in $\mathbb{F}_r$. Then, they provided the variant of Tate pairing

$$T_r(\overline{D_1}, \overline{D_2})^m = \left(f^{\lambda^3}_{\lambda,D_1}(D_2) \cdot f^{\lambda^2}_{\lambda,D_1}(\hat{\varphi}(D_2)) \cdot f^{\lambda}_{\lambda,D_1}(\hat{\varphi}^2(D_2)) \cdot f_{\lambda,D_1}(\hat{\varphi}^3(D_2)) \cdot u_1(D_2)\right)^{(p^k-1)/r}$$

where $\hat{\varphi}$ is the dual isogeny of φ. The loop length of each Miller function $\log_2 \lambda$ is around $(\log_2 r)/4$, therefore the pairing is *super-optimal* [90] nevertheless four Miller functions should be evaluated.

The detailed cost estimation of our Weil pairings with an automorphism will be described in the next chapter 5. It is clear that the cost of our pairing is considerably less than that of a Tate-type pairing with an automorphism for high security levels.

4.3 Summary

This chapter proposed a new variant of Weil pairing with an efficiently computable automorphism on a genus 2 curve. This result can be considered as a generalization of [95] in elliptic case.

Indeed, a exponentiation of the standard Weil pairing can be written a Weil-type pairing whose Miller loop lengths are bit size of an eigenvalue of the automorphism (Theorem 1).

Furthermore, an effective computation method of the Weil pairing with the automorphism is showed by Theorem 3. From this we can take reduced divisors for the inputs of the pairing and evaluate the Miller functions at the effective part of the divisor without performing any normalization for the Miller functions.

In addition, the construction of Weil-type pairing in the Hess-Vercauteren framework is proved by using the Weil pairing with an automorphism (Theorem 2).

5. The Pairings on the Kawazoe-Takahashi Families of Curves at the High Security Levels

This chapter provides the detailed cost evaluation of pairings on the Kawazoe-Takahashi families of curves. Chapter 3 showed that the optimum choice for an efficient pairing on genus 2 curve is Kawazoe-Takahashi cyclotomic families. Here an effective construction of the pairing is revealed by constructing optimal HV pairing, twisted HV pairing, and twisted Ate pairing explicitly. Then the cost of selected pairing is analyzed carefully in the Miller loop and the computation of final exponentiation. Furthermore, the cost of a Weil-type pairing with an automorphism described in chapter 4 is evaluated.

5.1 The Kawazoe-Takahashi Family of Curves with the Twists of Degree 8

In this section, we consider the Kawazoe-Takahashi families of genus 2 curves and pairings of different types on the curves. As described in the previous section, we are required to use twists of a high degree to reduce the cost of computing a pairing at high security levels. We therefore consider the following curves

$$C\colon y^2 = x^5 + ax,\ a \in \mathbb{F}_p,\ p \equiv 1 \pmod 8,$$

because such curves may have a twist of degree 8. In this paper, we assume that the embedding degree k is even. Indeed, we suppose that k is divisible by 8.

There is an efficiently computable automorphism φ induced by a morphism of C:

$$\begin{aligned} \varphi\colon C &\to C \\ P = (x, y) &\mapsto \varphi(P) = (\zeta_8^2 x, \zeta_8 y), \end{aligned}$$

where ζ_8 is a primitive 8-th root of unity and note that $\zeta_8 \in \mathbb{F}_p$. The morphism $\hat{\varphi}$ defined by $\hat{\varphi}(P) = (\zeta_8^{-2} x, \zeta_8^{-1} y)$ induces the dual isogeny of φ and it holds that $\varphi \circ \hat{\varphi} = [1]$. The characteristic polynomial of $\varphi, \hat{\varphi}$ is $T^4 + 1$ and φ and $\hat{\varphi}$ act as a multiplication map by $[\lambda], [\hat{\lambda}]$ on the group $\mathrm{Jac}_C(\mathbb{F}_p)[r]$ ($r^2 \nmid \sharp\mathrm{Jac}_C(\mathbb{F}_p)$), respectively, where λ and $\hat{\lambda}$ are roots of $T^4 + 1 \equiv 0 \pmod r$.

For this family of curves, we can take twists of degree d. Let $c \in \mathbb{F}_{p^e}$ be a l-th power non-residue, where $e = k/d$ and $l \mid d$; hence, the twist of C can be defined as:

$$C_t \colon y^2 = x^5 + acx,$$

and we have the isomorphism

$$\begin{aligned} \psi \colon C_t(\mathbb{F}_{p^k}) &\to C(\mathbb{F}_{p^k}) \\ P(x,y) &\mapsto \psi(P) = (c^{-\frac{2}{d}}x, c^{-\frac{5}{d}}y). \end{aligned}$$

This induces the automorphism $\psi \colon \mathrm{Jac}_{C_t}(\mathbb{F}_{p^k}) \to \mathrm{Jac}_C(\mathbb{F}_{p^k})$;

$$\begin{aligned} [x^2 + u_1'x + u_0', v_1'x + v_0'] &\mapsto [x^2 + c^{-\frac{2}{d}}u_1'x + c^{-\frac{4}{d}}u_0', c^{-\frac{3}{d}}v_1'x + c^{-\frac{5}{d}}v_0'], \\ [x + u_0', v_0'] &\mapsto [x + c^{-\frac{2}{d}}u_0', c^{-\frac{5}{d}}v_0'], \\ \mathcal{O} &\mapsto \mathcal{O} \end{aligned}$$

with the Mumford representation. Note that denominator elimination techniques can be used since $c^{-\frac{2}{d}} \in \mathbb{F}_{p^{k/2}}$.

Similarly, there is an efficiently computable automorphism φ' induced by

$$\begin{aligned} \varphi' \colon C_t &\to C_t \\ P' = (x', y') &\mapsto \varphi(P') = (\zeta_8^2 x', \zeta_8 y'), \end{aligned}$$

and the dual isogeny $\hat{\varphi}'$ derived from $\hat{\varphi}'(P') = (\zeta_8^{-2}x', \zeta_8^{-1}y')$.

For a divisor D, $\epsilon(D)$ denotes the effective part of reduced divisor $\rho(D)$. We also write that $D_1 \cap D_2 = \varnothing$ when divisors D_1 and D_2 have disjoint support.

5.2 For the 192-bit Security Level

Here we consider the Kawazoe-Takahashi family of hyperelliptic curves with embedding degree $k = 16$ for pairings at the 192-bit security level. This family of curves is explicitly given by using the cyclotomic polynomial and can be parametrized in a way that is convenient for constructing the optimal pairings in the Hess-Vercauteren framework.

It is possible to construct the cyclotomic family of type I [58, Section 6.1], where $(k, h) = (16, 5)$.

$$C \colon y^2 = x^5 + ax$$

is defined over $\mathbb{F}_p$, where $p = 1 \pmod 8$ such that the parameter p and r, which is a prime factor of the order of $\mathrm{Jac}_C(\mathbb{F}_p)$, are parametrized by $t \in \mathbb{Z}$ as follows:

$$r(t) = \Phi_{16}(t)/2 = (t^8 + 1)/2,$$
$$p(t) = (2t^{14} - 4t^{13} + 2t^{12} + t^{10} + 2t^9 + t^8 + 2t^6 + 4t^5 + 2t^4 + t^2 + 2t + 1)/8.$$

Hence, the ρ-value $\rho = g \log p / \log r$ is approximately 3.5 since $p \approx r^{14/8}$.

5.2.1 Optimal HV Pairing

The construction of optimal HV pairings requires us to choose an effective polynomial $h(x) = \sum_{i=0}^{n} h_i x^i \in \mathbb{Z}[x]$ such that $|h_i| \leqslant r^{1/\varphi(k)}$. Vercauteren showed several optimal HV pairings on elliptic curve families by finding the shortest vectors in a lattice [90]. Specifically, for a $\varphi(k)$-dimensional lattice (spanned by the rows)

$$L = \begin{pmatrix} r & 0 & 0 & \cdots & 0 \\ -p \pmod r & 1 & 0 & \cdots & 0 \\ -p^2 \pmod r & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ -p^{\varphi(k)-1} \pmod r & 0 & 1 & \cdots & 0 \end{pmatrix}, \tag{4}$$

he used the function `ShortestVectors()` or `ShortVectors()` in Magma [24] for specific input integers, and he found the parametrized shortest vectors by interpolating the parameters r and p for each input.

Since t should be an odd integer we substitute $t = 2x + 1$ in $r(t), p(t)$ and obtain

$$r(x) = 128x^8 + 512x^7 + 896x^6 + 896x^5 + 560x^4 + 224x^3 + 56x^2 + 8x + 1,$$
$$p(x) = 4096x^{14} + 24576x^{13} + 67584x^{12} + 112640x^{11} + 126848x^{10} + 102144x^9 + 61184x^8 + 28544x^7 + 11184x^6 + 4064x^5 + 1432x^4 + 456x^3 + 115x^2 + 20x + 2.$$

Now we can calculate the shortest vectors for the lattice L by using Magma and obtain the vector

$$\begin{aligned} V(x) &= [2x + 1, 0, 0, 0, 0, 1, 0, 0] \\ &= [t, 0, 0, 0, 0, 1, 0, 0]; \end{aligned}$$

therefore, it holds that $2x + 1 + p(x)^5 \equiv 0 \pmod{r(x)}$. We then compute the Miller function except for the final exponentiation of the HV pairing $a_{p,h}$ as

$$f_{t+p^5,D_2} = f_{t,D_2} f_{p^5,D_2} \cdot g_{[t]D_2,[p^5]D_2},$$

where

$$\mathrm{div}\left(g_{[t]D_2,[p^5]D_2}\right) = [t]D_2 + [p^5]D_2 - [t+p^5]D_2.$$

Now we consider the Frobenius eigenspace $\mathbb{G}_1$ and $\mathbb{G}_2$ as the domain of the pairing; thus, it holds that $f_{p^5,D_2} = f_{1,D_2}^{p^5}$ and f_1 is constant and we can therefore write

$$a_{p,h}(D_2, D_1) = f_{t,D_2} \cdot g_{[t]D_2,[p^5]D_2}(D_1)^{(q^{16}-1)/r}. \tag{5}$$

5.2.2 Twisted HV Pairing

Now C has a twist of degree $d = 8$ defined over $\mathbb{F}_{p^2}$ as follows:

$$\begin{aligned} C_t &\colon y^2 = x^5 + acx, \\ \psi &\colon C_t(\mathbb{F}_{p^{16}}) \to C(\mathbb{F}_{p^{16}}) \\ &\quad (x, y) \mapsto (c^{\frac{1}{4}}x, c^{\frac{5}{8}}y), \end{aligned} \tag{6}$$

where $c \in \mathbb{F}_{p^2}$ is not the l-th power residue in $\mathbb{F}_{p^2}$ for $l \in \{1, 2, 4, 8\}$.

In our case, since $m = \gcd(k, d) = 8$ and $e = k/m = 2$ we can represent $\mathbb{G}_2$ as

$$\mathbb{G}_2 = \mathrm{Jac}_C(\mathbb{F}_p^k)[r] \cap \ker\left([\xi_m]\pi^2 - 1\right).$$

This necessitates the search for shorter vectors for $h(x)$, where the coefficients of p^i (i: odd) are set equal to 0 to reduce the Miller function in the same manner as HV pairings. For the lattice

$$L = \begin{pmatrix} r & 0 & 0 & 0 \\ -p^2 \pmod r & 1 & 0 & 0 \\ -p^4 \pmod r & 0 & 1 & 0 \\ -p^6 \pmod r & 0 & 0 & 1 \end{pmatrix},$$

we can find the vector

$$W(x) = [(2x+1)^2, 1, 0, 0] = [t^2, 1, 0, 0]$$

by using `ShortVectors()` and it holds $(2x+1)^2 + p(x)^2 \equiv 0 \pmod{r(x)}$. In this case, the length of the Miller loop is twice that of optimal pairing. The twisted HV pairing can be computed as follows:

$$a^{\text{twist}}_{p^2,h}(D_1, D_2) = f_{t^2,D_1} \cdot g_{[t^2]D_1,[p^2]D_1}(D_2)^{(q^{16}-1)/r}. \tag{7}$$

5.2.3 Twisted Ate Pairing

Zhang [92] proposed hyperelliptic twisted Ate pairing. Here we confirm that previous twisted HV pairing in 5.2.2 corresponds to a twisted Ate pairing. Zhang showed that

$$\begin{aligned} a^{\text{twist}} \colon \mathbb{G}_1 \times \mathbb{G}_2 &\to \mu_r \\ (\overline{D_1}, \overline{D_2}) &\mapsto f_{q^{ei} \pmod r, D_1}(D_2)^{(q^k-1)/r} \end{aligned}$$

is a bilinear pairing [92, Theorem 4], where e is the same as described in 5.2.2. As we aim to use the smallest $p^{ei} \pmod r$, it holds that $p^{10} \pmod r = t^2$. Therefore, we can simply compute

$$a^{\text{twist}}(D_1, D_2) = f_{t^2,D_1}(D_2)^{(q^{16}-1)/r}. \tag{8}$$

No additional rational function occurred in the twisted HV pairing in the previous subsection.

5.3 For the 256-bit Security Level

In this subsection, we describe the explicit construction of the Kawazoe-Takahashi cyclotomic family of curves with an embedding degree $k = 24$ for pairings on them at the 256-bit security level. We use these methods for generating the cyclotomic family of type II [58, Section 6.2], where $(k, h) = (24, 11)$.

This is similar to the case $k = 16$ in which these curves

$$C \colon y^2 = x^5 + ax$$

are defined over $\mathbb{F}_p$, where $p = 1 \pmod 8$. Then $r(t)$ and $p(t)$ are given as follows:

$$\begin{aligned} r(t) &= \Phi_{24}(t) = t^8 - t^4 + 1, \\ p(t) &= (2t^{12} + 4t^{11} + 3t^{10} - 2t^9 - t^8 + 4t^7 - 3t^6 + 2t^5 + t^4 - 4t^3 + 3t^2 - 2t + 1)/8 \end{aligned}$$

We then have that $\rho \approx 3$.

5.3.1 Optimal HV Pairing

Since $t = 1 \pmod 4$ is equivalent to

$$\tilde{p}(t) = 2t^{12}+4t^{11}+3t^{10}-2t^9-t^8+4t^7-3t^6+2t^5+t^4-4t^3+3t^2-2t+1 = 0 \pmod 8,$$

we substitute $t = 4x + 1$ in $r(t), p(t)$ and obtain

$$\begin{aligned} r(x) &= 65536x^8 + 131072x^7 + 114688x^6 + 57344x^5 + 17664x^4 + 3328x^3 \\ &\quad + 352x^2 + 16x + 1, \\ p(x) &= 4194304x^{12} + 14680064x^{11} + 23461888x^{10} + 22544384x^9 + 14467072x^8 \\ &\quad + 6529024x^7 + 2127360x^6 + 505344x^5 + 87168x^4 + 10688x^3 + 886x^2 + 44x + 1. \end{aligned}$$

In a way similar to that described in Section 5.2.1, we obtain one of the shortest vectors for the lattice L (4)

$$\begin{aligned} V(x) &= [1, 4x + 1, 0, 0, 0, 0, 0, 0] \\ &= [1, t, 0, 0, 0, 0, 0, 0], \end{aligned}$$

where $1 + (4x + 1)p(x) \equiv 0 \pmod{r(x)}$. We then have the optimal HV pairing

$$a_{p,h}(D_2, D_1) = f^p_{t,D_2} \cdot g_{[1]D_2,[tp]D_2}(D_1)^{(q^{24}-1)/r}. \tag{9}$$

5.3.2 Twisted HV Pairing

The curve C with $k = 24$ has a twist of degree 8, which is the same form of (6) in Section 5.2.2 where $m = \gcd(k, d) = 8$ and $e = k/m = 3$. Then the group $\mathbb{G}_2$ can be represented as

$$\mathbb{G}_2 = \mathrm{Jac}_C(\mathbb{F}^k_p)[r] \cap \ker([\xi_m]\pi^3 - 1).$$

We then compute short vectors for the lattice

$$L = \begin{pmatrix} & r & 0 & 0 \\ -p^3 & \pmod r & 1 & 0 \\ -p^6 & \pmod r & 0 & 1 \end{pmatrix},$$

and found

$$W(x) = [(4x + 1)^2, 4x + 1, 1] = [t^2, t, 1],$$

which is not the shortest vector but has coefficients with limited efficiency h_i. Then we obtain a twisted HV pairing

$$a^{\text{twist}}_{p^3,h}(D_1, D_2) = f_{t^2,D_1} \cdot f^{p^3}_{t,D_1} \cdot g_{[t^2]D_1,[tp^3+p^6]D_1} \cdot g_{[tp^3]D_1,[p^6]D_1}(D_2)^{(q^{24}-1)/r}. \quad (10)$$

5.3.3 Twisted Ate Pairing

The case of $p^{ei} \pmod r$ requires us to take $p^9 = t^3 \pmod r$. Then the twisted Ate pairing is given as follows:

$$a^{\text{twist}}(D_1, D_2) = f_{t^3,D_1}(D_2)^{(q^{24}-1)/r}. \quad (11)$$

5.4 Cost Estimation

In this section, we provide cost estimates of the pairings on the Kawazoe-Takahashi curve of embedding degree 16 and 24 as described in Section 5.2 and 5.3, respectively.

In each case, here we focus on the twisted Ate pairing and the Weil pairing with automorphism. For the three pairings of the Tate type, i.e., the optimal HV pairing, twisted HV pairing, and twisted Ate pairing, we conclude that the twisted Ate pairing is the fastest. In contrast to [2] in which the authors evaluated the optimal HV pairings on elliptic curves, the cost of arithmetic on Jac_C is considerably higher than that on elliptic curves. Hence, this necessitates the selection to use divisors defined over base field $\mathbb{F}_p$ as the first input of the pairing. Indeed, for $k = 16$ the optimal pairing can be computed as

$$\begin{aligned} a_{p,h}(\psi(D_2'), D_1) &= f_{t,\psi(D_2')} \cdot g_{[t]D_2,[p^5]D_2}(D_1)^{(q^{16}-1)/r} \\ &= f_{t,D_2'} \cdot g_{[t]D_2,[p^5]D_2}(\psi^{-1}(D_1))^{(q^{16}-1)/r}, \end{aligned}$$

where $D_2' \in \text{Jac}_{C_t}(\mathbb{F}_{p^2})$. Although the length of the Miller loop is half of that of the twisted pairings, the total cost can be expected to be higher with the cost of arithmetic on Jac_{C_t} over the quadratic extension field $\mathbb{F}_{p^2}$.

5.4.1 Notations and Cost of Miller Algorithm

We denote the cost of a multiplication and a squaring in $\mathbb{F}_{p^i}$ by M_i and S_i, respectively , and I_i denotes the cost of an inversion in $\mathbb{F}_{p^i}$. We also suppose that the cost of a squaring is equal to that of a multiplication in $\mathbb{F}_p$, i.e., $M_1 = S_1$.

We compute $f_{s,D_1}(\epsilon(\psi(D_2')))$ with the standard Miller's algorithm (Algorithm 1 in Section 3.2).

We perform arithmetic efficiently on $\mathrm{Jac}(\mathbb{F}_{p^i})$ with C and C_t by using the explicit formula and the dedicated coordinate system given by [33]. For a curve

$$y^2 = f(x),\ f(x) = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0,$$

where $f_2, f_3 = 0$, a doubling requires $35M_i + 5S_i$, and this enables us to perform a mixed addition with $36M_i + 5S_i$ by [33, Section 4.6].

Here we denote the cost of each computation by T_* as in [33] and [74] as follows:

- T_D: the cost of doubling a non-degenerate divisor,
- T_A: the cost of adding two non-degenerate divisors,
- T_G: the cost of evaluating the line function G at the divisor $\epsilon(\psi(D_2'))$,
- T_{sk}, T_{mk}: the cost of squaring and multiplication (respectively) in $\mathbb{F}_{p^k}$ in each Miller step,
- T_{fe}: the cost of performing the final exponentiation.

We can then write the cost of evaluating the Miller function $f_{\lambda,D_1}(\epsilon(\psi(D_2')))$ (without including the final exponentiation) for a reduced divisor D_2' by

$$\mathbf{D}(T_D + T_G + S_k + bM_k) + \mathbf{A}(T_A + T_G + bM_k),$$

where $\mathbf{D}$ and $\mathbf{A}$ denote the number of doublings and additions in Jac_C for the Miller loop, respectively, and $b = \deg(\epsilon(D_2'))$.

5.4.2 Cost of Evaluating a Line Function

Here we reveal the cost of evaluating the line function T_G. In Cantor's algorithm in Miller's algorithm 1, we need to evaluate the rational function by substituting the points associated with $\psi(D_2')$. The rational function can be obtained as

$$\frac{y - v(x)}{u'(x)},$$

where the degree of $v(x)$ is at most 3 [38]. Since the x-coordinate of $\psi(D_2')$ is defined in $\mathbb{F}_{p^{k/2}}$, we can use denominator elimination; thus, we need not evaluate $u'(x)$. The new coordinate system from [33] should be used to evaluate

$$c_D(x,y) = (\tilde{r}z_{11})y - ((s_1'z_{11})x^3 + l_2x^2 + l_1x + l_0),$$
$$c_A(x,y) = (\tilde{r}z_{21})y - ((s_1'z_{21})x^3 + l_2x^2 + l_1x + l_0)$$

for a doubling and an addition, respectively, instead of $y - v(x)$, where the parameters are specified in [32, Table 4,5].

Let f be the intermediate pairing value when we use the effective part of reduced divisors $\epsilon(\psi(D_2'))$ as second inputs for the pairing, where $D_2' = \sum_{i=1}^{j}(x_i, y_i) - b_lP_\infty,\ l \in \{1,2\}$. In each doubling step we compute

$$f^2 \cdot c_D(\epsilon(\psi(D_2'))) = f^2 \cdot \prod_{i=1}^{l} c_D(c^{-\frac{2}{d}}x_i, c^{-\frac{5}{d}}y_i),$$

and

$$f \cdot c_A(\epsilon(\psi(D_2'))) = f \cdot \prod_{i=1}^{l} c_A(c^{-\frac{2}{d}}x_i, c^{-\frac{5}{d}}y_i)$$

in each addition step.

After precomputing $(c^{-\frac{2}{d}}x_i)^2,\ (c^{-\frac{2}{d}}x_i)^3$ with $l(S_{k/2} + M_{k/2})$, we can compute each of $c_D(\psi(D_2'))$ and $c_A(\psi(D_2'))$ with $T_G = l(M_k' + 3 \cdot M_{k/2}') + (l-1)M_k$, where M_s' denotes the cost of a sparse multiplication ab, where $a \in \mathbb{F}_p$ and $b \in \mathbb{F}_{p^s}$.

In our case, D_2' is defined over the extension field $\mathbb{F}_{p^e}\backslash\mathbb{F}_p$; hence, we use a degenerate divisor for D_2' to reduce the cost of pairings.

5.4.3 Cost of Pairings on the Kawazoe-Takahashi Curve with $k = 16$

Explicit Parameters

We take the following curve

$$C\colon y^2 = x^5 + 7x,$$

where

$$
\begin{aligned}
r &= 2032576060946361904940007374704524749046164908931845007118942296515047\backslash \\
&\quad 7383380813830984495803858031945687463818769531 3 \text{ (384 bits)}, \\
p &= 5173976026423507901566639372000370573652256115110390892678668078429788\backslash \\
&\quad 4261101799277683689529883901978391236286542183439768741027536766273 5\backslash \\
&\quad 0494517372518535753335875110267591189392701991247446900148883577 \\
&\quad (671 \text{ bits}), \\
t &= 282575562145795 = 2^{48} + 2^{40} + 2^{30} + 2^{16} + 2 + 1
\end{aligned}
$$

for the 192-bit security level. Since $p = 5 \pmod{12}$, $3 \in \mathbb{F}_p$ is a quadratic non-residue. Hence, we can construct $\mathbb{F}_{p^{16}}$ as a 1-2-4-8-16 quadratic tower extension field

$$
\begin{aligned}
\mathbb{F}_{p^2} &\simeq \mathbb{F}_p[x]/(x^2 - 3), \\
\mathbb{F}_{p^4} &\simeq \mathbb{F}_{p^2}[y]/(y^2 - \alpha), \ (\alpha^2 - 3 = 0), \\
\mathbb{F}_{p^8} &\simeq \mathbb{F}_{p^4}[z]/(z^2 - \beta), \ (\beta^2 - \alpha = 0), \\
\mathbb{F}_{p^{16}} &\simeq \mathbb{F}_{p^8}[s]/(s^2 - \gamma), \ (\gamma^2 - \beta = 0).
\end{aligned}
$$

Table 3 lists the costs of basic arithmetic and Frobenius operations in the extension fields.

Table 3. Costs of arithmetic in the extension fields for $k = 16$

Field	Multiplication	Squaring	Inversion
$\mathbb{F}_{p^2}$	$3M_1$	$2M_1$	I_2
$\mathbb{F}_{p^4}$	$9M_1$	$6M_1$	I_4
$\mathbb{F}_{p^8}$	$27M_1$	$18M_1$	I_8
$\mathbb{F}_{p^{16}}$	$81M_1$	$54M_1$	$I_1 + 134M_1$
$G_{\Phi_2(p^8)}$	$81M_1$	$2M_8 = 36M_1$ [46]	Conjugation

Operation	Cost
$p/p^3/p^5/p^7$-Frobenius	$14M_1$
p^2/p^6-Frobenius	$12M_1$
p^4-Frobenius	$8M_1$

Twisted Ate Pairing

Here we consider the cost of the twisted Ate pairing (8) described in Section 5.2.3 where D_1 is non-degenerate and D_2' is degenerate. We first evaluate the Miller loop. Since

$$\begin{aligned} t^2 &= 79848948322012052144836182025 \\ &= 2^{96} + 2^{89} + 2^{80} + 2^{79} + 2^{71} + 2^{65} + 2^{60} + 2^{57} + 2^{50} + 2^{49} + 2^{47} + 2^{42} + 2^{41} \\ &\quad + 2^{33} + 2^{31} + 2^{18} + 2^{17} + 2^3 + 1 \end{aligned}$$

we have $\mathbf{D} = 96$ and $\mathbf{A} = 18$.

The cost of precomputation in Section 5.4.2 is $S_8 + M_8 = 45M_1$. Furthermore, we have that $T_G = 16M_1 + 3 \cdot 8M_1 = 40M_1$ with the precomputation. Hence, the computation of the Miller loop requires

$$45M_1 + 96(35M_1 + 5S_1 + 40M_1 + S_{16} + M_{16}) + 18(36M_1 + 5S_1 + 40M_1 + M_{16}) = 23601M_1.$$

Computation of the final exponentiation $f^{(p^{16}-1)/r}$ requires us to reduce the cost of the hard part $\Phi_{16}(p)/r$, where

$$\begin{aligned} (p^{16} - 1)/r &= (p-1)(p+1)(p^2+1)(p^4+1)(p^8+1)/r \\ &= (p^8 - 1) \cdot \frac{\Phi_{16}(p)}{r}. \end{aligned}$$

The use of the parametrization of $p(x)$ and $r(x)$ enables us to compute the coefficients of the following polynomial as elements in $\mathbb{Z}[x]$.

$$\frac{(p(x)^8 + 1)}{r(x)} = \sum_{i=0}^{7} l_i(x) p(x)^i,$$

where

$$
\begin{aligned}
l_0(x) &= 256x^9 + 896x^8 + 1344x^7 + 1120x^6 + 568x^5 + 196x^4 + 66x^3 + 27x^2 + 8x + 3 \\
l_1(x) &= -2048x^{12} - 10240x^{11} - 23040x^{10} - 30720x^9 - 26944x^8 - 16448x^7 \\
&\quad - 7408x^6 - 2752x^5 - 980x^4 - 348x^3 - 111x^2 - 26x - 3 \\
l_2(x) &= -64x^7 - 160x^6 - 160x^5 - 80x^4 - 22x^3 - 7x^2 - 4x - 1 \\
l_3(x) &= 512x^{10} + 2048x^9 + 3584x^8 + 3584x^7 \\
&\quad + 2256x^6 + 960x^5 + 328x^4 + 120x^3 + 43x^2 + 14x + 3 \\
l_4(x) &= -4096x^{13} - 22528x^{12} - 56320x^{11} - 84480x^{10} - 84608x^9 - 59840x^8 \\
&\quad - 31264x^7 - 12912x^6 - 4712x^5 - 1676x^4 - 570x^3 - 163x^2 - 32x - 3 \\
l_5(x) &= -128x^8 - 384x^7 - 480x^6 - 320x^5 - 124x^4 - 36x^3 - 15x^2 - 6x - 1 \\
l_6(x) &= 1024x^{11} + 4608x^{10} + 9216x^9 + 10752x^8 + 8096x^7 + 4176x^6 + 1616x^5 \\
&\quad + 568x^4 + 206x^3 + 71x^2 + 20x + 3 \\
l_7(x) &= 32x^6 + 64x^5 + 48x^4 + 16x^3 + 3x^2 + 2x + 1.
\end{aligned}
$$

We then simplify the polynomials as follows:

$$
\begin{aligned}
l_0(x) &= (x+1)A(x), \\
l_1(x) &= -(x+1)(2x+1)^3A(x), \\
l_2(x) &= -(2x+1)B(x), \\
l_3(x) &= (x+1)(2x+1)A(x), \\
l_4(x) &= -(x+1)(2x+1)^4A(x), \\
l_5(x) &= -(2x+1)^2B(x), \\
l_6(x) &= (x+1)(2x+1)^2A(x), \\
l_7(x) &= B(x)
\end{aligned}
$$

where

$$
\begin{aligned}
A(x) &= 256x^8 + 640x^7 + 704x^6 + 416x^5 + 152x^4 + 44x^3 + 22x^2 + 5x + 3, \\
B(x) &= 32x^6 + 64x^5 + 48x^4 + 16x^3 + 3x^2 + 2x + 1, \\
A(x) &= (8x^2 + 4x + 2)B(x) - 32x^5 - 32x^4 - 16x^3 - 3x + 1.
\end{aligned}
$$

The value f is determined after computing the Miller loop by computing $g = f^{p^8-1}$ with $M_{16} + I_{16}$. Note that g is an element of the cyclotomic subgroup $G_{\Phi_2(p^8)}$. Since $t = 2x + 1$,

$$x = 2^{47} + 2^{39} + 2^{29} + 2^{15} + 1.$$

Hence, computing h^x requires 47 cyclotomic squarings and 4 multiplications in $F_{p^{16}}$ for an element $h \in G_{\Phi_2(p^8)}$.

Let S_c denote the cost of a cyclotomic squaring $G_{\Phi_2(p^8)}$. We first compute $g^{B(x)}$ with $304S_c + 32M_{16}$. We can then compute $g^{A(x)}$ with $104S_c + 16M_{16}$ by using the results we obtained for computing $g^{B(x)}$. After that we compute

$$\begin{aligned} g^{(2x+1)B(x)} &\to g^{(2x+1)^2B(x)}, \\ g^{(x+1)A(x)} &\to g^{(x+1)(2x+1)A(x)} \\ &\to g^{(x+1)(2x+1)^2A(x)} \to g^{(x+1)(2x+1)^3A(x)} \to g^{(x+1)(2x+1)^4A(x)} \end{aligned}$$

with $335S_c + 35M_{16}$ to obtain g^{l_i}. The computation of $(g^{l_i})^{p^i}$ requires $88M_1$ and we can compute $g^{(p^8+1)/r} = \prod_{i=0}^{7} g^{l_1 p^i}$ with $7M_{16}$.

All in all, the cost of performing the final exponentiation is

$$M_{16} + I_{16} + 743S_c + 90M_{16} + 81M_1 = 34341M_1 + I_1.$$

We next consider the method by Scott et al. [83] by computing a multi-exponentiation with a vectorial chain. We first need to compute g^{x^i} $(1 \leqslant i \leqslant 13)$ with $13(47S_c + 4M_{16}) = 611S_c + 52M_{16}$. Second, we compute $(g^{x^i})^{p^j}$ with $(40 \cdot 14 + 20 \cdot 12 + 14 \cdot 8)M_1 = 912M_1$ for the coefficients of $l_i(x)$ as described in [83, Section 5]. Finally, it is necessary compute a vectorial addition chain such as [83, Section 5] to compute the multi-exponentiation. This requires us to compute an addition chain from coefficients set from $l_i(x)$, and we obtain

[1, 2, 3, 4, 6, 7, 8, 14, 15, 16, 20, 22, 26, 27, 32, 36, 43, 48, 64, 66, 68, 71, 80, 111, 112, 120, 124, 128, 160, 163, 196, 206, 256, 320, 328, 348, 384, 480, 512, 520, 568, 570, 896, 960, 980, 1024, 1120, 1344, 1348, 1360, 1616, 1676, 2048, 2256, 2272, 2752, 3072, 3584, 3592, 4096, 4608, 4656, 4712, 4716, 7408, 7528, 8096, 8352, 9216, 10240, 10752, 10864, 11776, 12912, 16448, 16704, 22528, 23040, 26944, 27968, 28352, 30720, 30752, 31264, 31872, 53760, 56320, 59840, 84480, 84608].

We then compute a vectorial addition chain of length 230 from this chain. This implies that $230 - 71 = 159$ multiplications in $\mathbb{F}_{p^{16}}$ including 3 cyclotomic squarings are required to obtain the desired value, where 71 is the number of unit vectors. Performing the final exponentiation with the vectorial chain requires

$$M_{16} + I_{16} + 611S_c + 52M_{16} + 912M_1 + 3S_c + 156M_{16} = 39167M_1 + I_1,$$

which is higher than the cost we evaluated above.

Consequently, the estimated cost of the twisted Ate pairing is $57942M_1 + I_1$.

Weil Pairing with Automorphism

Here we evaluate the cost of Weil pairing with automorphism φ (3) in Theorem 3 for both a non-degenerate divisor D_1 and a degenerate divisor D_2'.

We need to evaluate the two Miller functions

$$f_{\lambda,D_1}(\epsilon(\psi(D_2'))) \text{ and } f_{\lambda,D_2'}(\epsilon(\psi^{-1}(D_1)))$$

naively; however, we can use some efficient methods by using the feature of the quadratic extension field $\mathbb{F}_{p^k}/\mathbb{F}_{p^{k/2}}$.

First, it is not necessary to process the two variables for the Miller functions individually in each Miller step; instead, we can share them as described at the end of Section 3 of [95]. In addition, we can replace the computation of the inverse of an element $a \in \mathbb{F}_{p^k}$ by taking its conjugate $\overline{a}$ when performing the exponentiation $a^{p^{k/2}-1}$. Indeed, as described in [80, Section 6], for $a + b\alpha \in \mathbb{F}_{p^k}$ we have that

$$(1/(a + b\alpha))^{p^{k/2}-1} = (a - b\alpha)^{p^{k/2}-1} = (\overline{a + b\alpha})^{p^{k/2}-1}.$$

Hence, we can obtain a conjugate of the evaluated value in each Miller step instead of an inverse.

Let T_G' denote the cost of evaluating a line function G at the divisor $\psi^{-1}(D_1)$. The cost of the precomputation for T_G' is the same as for T_G, that is, $l(S_{k/2}+M_{k/2})$. We can then compute each of $c_D(\psi^{-1}(D_1))$ and $c_A(\psi^{-1}(D_1))$ with $T_G' = l(M_k'' + 3 \cdot M_{k/2}'') + (l-1)M_k$, where M_s'' denotes the cost of a sparse multiplication ab, where $a \in \mathbb{F}_{p^e}$ and $b \in \mathbb{F}_{p^s}$.

We then have that $T_G = 40M_1$ and $T'_G = 201M_1$ with each precomputation. Hence, in each Miller step we compute

$$f^2 \cdot c_D\psi(D'_2) \cdot \overline{c_D(\psi^{-1}(D_1))}$$

with $T_G + T'_G + S_{16} + 2M_{16} = 457M_1$ and

$$f \cdot c_A\psi(D'_2) \cdot \overline{c_A(\psi^{-1}(D_1))}$$

with $T_G + T'_G + 2M_{16} = 403M_1$.

Fan et al. [32, Section 5.1] described a method to select λ, which has a low Hamming weight. Unfortunately, we cannot search for an appropriate λ when $\lambda^4 + 1 = 0 \pmod{r}$ because of the cost of factorization of $\lambda^4 + 1$. In the case $k = 16$, we can select λ for the parameters described above such that

$$\lambda = 79848948322012052144836182025 = t^2.$$

Indeed, it holds that $(t^2)^4 + 1 = t^8 + 1 = 2r$. Hence, we have $\mathbf{D} = 96$ and $\mathbf{A} = 18$.

All in all, the cost of Weil pairing is

$$\begin{aligned} &135M_1 + 96 \cdot (40 + 115 + 457)M_1 + 18 \cdot (41 + 118 + 403)M_1 + M_{16} + I_{16} \\ &= 69218M_1 + I_1. \end{aligned}$$

5.4.4 Cost of Pairings on the Kawazoe-Takahashi Curve with $k = 24$

Explicit Parameters

For pairings at the 256-bit security level, we use the following curve

$$C\colon y^2 = x^5 + 25x,$$

where

$$\begin{aligned} r = {} & 220988771507618765439544939273027574632842847086999714362434465575909\backslash \\ & 636693573636526934272222972567467452755324884009612876213656017295278\backslash \\ & 57522838549299201 \text{ (513 bits)}, \\ p = {} & 821287702757071793233174823374911914277690697528659480300274611170050\backslash \\ & 073854437470732134626137074250773125583100256581431480191558956819400\backslash \\ & 427433086864108325673035786183442016893913625168696918275541760805525\backslash \\ & 97332126073552131797811 3 \text{ (768 bits)}, \\ t = {} & 19635694925091176449 = 2^{64} + 2^{60} + 2^{55} + 2^{39} + 1 \end{aligned}$$

We ensure efficiency by using methods of squarings in the cyclotomic subgroup $G_{\Phi_6(p^4)}$; hence, we assume that $p = 1 \pmod 6$. In addition, we have that $p = 1 \pmod 8$ and $p = 3 \pmod 5$. Hence, $5 \in \mathbb{F}_p$ is a quadratic nonresidue. In this case, we can construct $\mathbb{F}_{p^{24}}$ as $\mathbb{F}_{p^{24}} \simeq \mathbb{F}_p[x]/(x^{24} - 5)$. We aim to achieve efficient arithmetic on the fields by constructing the first extension field as $\mathbb{F}_{p^2} = \mathbb{F}_p[x]/(x^2 - 5)$, and define input divisor $D_2' \in \mathrm{Jac}_{C_t}$ over $\mathbb{F}_{p^3}$ as a requirement. Therefore, we consider $\mathbb{F}_{p^{24}}$ as the field constructed by 1-3-6-12-24 tower extension field only when we select an input divisor D_2'. Subsequently, we perform arithmetic in $\mathbb{F}_{p^{24}}$ constructed as an 1-2-4-12-24 tower extension field. Because $\mathbb{F}_{p^{24}} = \mathbb{F}_p(5^{1/24})$, it enables us to perform the basis conversion without additional costs for multiplications.

Table 4 contains the costs of basic arithmetic and Frobenius operations in the extension fields.

Table 4. Costs of arithmetic in the extension fields for $k = 24$

Field	Multiplication	Squaring	Inversion
$\mathbb{F}_{p^2}$	$3M_1$	$2M_1$	I_2
$\mathbb{F}_{p^3}$	$6M_1$	$5M_1$	I_3
$\mathbb{F}_{p^4}$	$9M_1$	$6M_1$	I_4
$\mathbb{F}_{p^{12}}$	$54M_1$	$36M_1$	I_8
$\mathbb{F}_{p^{24}}$	$162M_1$	$108M_1$	$I_1 + 275M_1$
$G_{\Phi_4(p^6)}$	$162M_1$	$6M_4 = 54M_1$ [46]	Conjugation

Operation	Cost
$p/p^3/p^5/p^7$-Frobenius	$22M_1$
p^2/p^6-Frobenius	$12M_1$
p^4-Frobenius	Conjugation

Twisted Ate Pairing

In a way similar to that in Section 5.4.3, we evaluate the twisted Ate pairing (11) in Section 5.3.3. For t^3, where t is specified as above, we have $\mathbf{D} = 193$ and $\mathbf{A} = 52$.

The precomputation in Section 5.4.2 requires $S_{12} + M_{12} = 90M_1$. We then

have that $T_G = 24M_1 + 3 \cdot 12M_1 = 60M_1$ with the precomputation. Hence, the computation of the Miller loop requires

$$90M_1 + 193(35M_1 + 5S_1 + 60M_1 + S_{24} + M_{24}) + 52(36M_1 + 5S_1 + 60M_1 + M_{24}) = 85176M_1.$$

For the exponent

$$\begin{aligned}(p^{24} - 1)/r &= (p^4 + p^2 + 1)(p^8 - 1)(p^8 - p^4 + 1)/r \\ &= (p^4 + p^2 + 1)(p^8 - 1)\frac{\Phi_{24}(p)}{r},\end{aligned}$$

we evaluate the hard part $\Phi_{24}(p)/r$ as follows. For the parametrized parameters $p(x)$ and $r(x)$, we write

$$\frac{(p(x)^8 - p(x)^4 + 1)}{r(x)} = \sum_{i=0}^{7} l_i(x)p(x)^i,$$

where

$$\begin{aligned}
l_0(x) &= 256x^5 + 448x^4 + 312x^3 + 102x^2 + 16x + 2 \\
l_1(x) &= -1024x^6 - 2048x^5 - 1696x^4 - 720x^3 - 166x^2 - 24x - 2 \\
l_2(x) &= 4096x^7 + 9216x^6 + 8832x^5 + 4576x^4 + 1384x^3 + 262x^2 + 32x + 2 \\
l_3(x) &= -16384x^8 - 40960x^7 - 44544x^6 - 27136x^5 - 10112x^4 - 2432x^3 - 390x^2 \\
&\quad - 40x - 2 \\
l_4(x) &= 65536x^9 + 180224x^8 + 219136x^7 + 153088x^6 + 67328x^5 + 19392x^4 + 3680x^3 \\
&\quad + 448x^2 + 32x \\
l_5(x) &= -262144x^{10} - 786432x^9 - 1056768x^8 - 831488x^7 - 422400x^6 - 144896x^5 \\
&\quad - 34112x^4 - 5472x^3 - 576x^2 - 32x \\
l_6(x) &= 1048576x^{11} + 3407872x^{10} + 5013504x^9 + 4382720x^8 + 2521088x^7 \\
&\quad + 1001984x^6 + 281344x^5 + 56000x^4 + 7776x^3 + 704x^2 + 32x \\
l_7(x) &= 64x^4 + 96x^3 + 54x^2 + 12x + 1
\end{aligned}$$

Then the polynomials can be simplified as follows:

$$\begin{aligned}
l_0(x) &= 2A(x),\\
l_1(x) &= -2(4x+1)A(x),\\
l_2(x) &= 2(4x+1)^2A(x),\\
l_3(x) &= -2(4x+1)^3A(x),\\
l_4(x) &= 32x(2x+1)A(x)B(x),\\
l_5(x) &= -32x(2x+1)(4x+1)A(x)B(x),\\
l_6(x) &= 32x(2x+1)(4x+1)^2A(x)B(x),\\
l_7(x) &= B(x)
\end{aligned}$$

where

$$\begin{aligned}
A(x) &= 128x^5 + 224x^4 + 156x^3 + 51x^2 + 8x + 1,\\
B(x) &= 64x^4 + 96x^3 + 54x^2 + 12x + 1,\\
A(x) &= 2xB(x) + 32x^4 + 48x^3 + 27x^2 + 6x + 1.
\end{aligned}$$

For the value f after computing the Miller loop, we can compute

$$g = f^{(p^4+p^2+1)(p^8-1)}$$

with $12M_1 + 3M_{24} + I_{24} = 773M_1 + I_1$. We now have $t = 4x + 1$, where

$$x = 2^{62} + 2^{58} + 2^{53} + 2^{37}.$$

Therefore, computing h^x requires $62S_c + 3M_{24}$ for an element $h \in G_{\Phi_2(p^8)}$, where S_c denotes the cost of a cyclotomic squaring $G_{\Phi_6(p^4)}$.

In the same manner as described in Section 5.4.3, we first compute $g^{B(x)}$ with $271S_c + 21M_{24}$. We can then compute $g^{A(x)}$ with $63S_c + 8M_{24}$ by using the results to compute $g^{B(x)}$. Again, this requires $271S_c + 21M_{24}$ for computing $g^{A(x)B(x)} = \left(g^{A(x)}\right)^{B(x)}$. After that we compute

$$\begin{aligned}
g^{(4x+1)A(x)} &\to g^{(4x+1)^2A(x)} \to g^{(4x+1)^3A(x)},\\
g^{(2x+1)A(x)B(x)} &\to g^{(2x+1)(4x+1)A(x)B(x)} \to g^{(2x+1)(4x+1)^2A(x)B(x)}
\end{aligned}$$

in order to obtain g^{l_i}. This costs $451S_c + 27M_{24}$. The computation of $(g^{l_i})^{p^i}$ requires $112M_1$ and we can compute $g^{(p^8-p^4+1)/r} = \prod_{i=0}^{7} g^{l_1 p^i}$ with $7M_{24}$.

All in all, the cost of performing the final exponentiation is

$$773M_1 + I_1 + 1056S_c + 84M_{24} + 112M_1 = 71517M_1 + I_1.$$

We next consider the method with a multi-exponentiation in the same manner when $k = 16$. We first compute g^{x^i} $(1 \leqslant i \leqslant 11)$ with $11(62S_c + 3M_{16}) = 682S_c + 33M_{16}$. Second, we compute $(g^{x^i})^{p^j}$ with $(31 \cdot 22 + 19 \cdot 12)M_1 = 910M_1$. We then find the following addition chain from the coefficients set from $l_i(x)$

[1, 2, 3, 4, 10, 12, 14, 16, 24, 32, 40, 54, 64, 78, 96, 102, 166, 178, 256, 258, 262, 270, 312, 314, 390, 448, 576, 704, 720, 1024, 1126, 1384, 1696, 1712, 1968, 2048, 2432, 3680, 3776, 4096, 4128, 4576, 5472, 5536, 6848, 6976, 7776, 7808, 7832, 8552, 8832, 9216, 10112, 10176, 16384, 16960, 19200, 19392, 27136, 34112, 34368, 38912, 40960, 44544, 48224, 56000, 56704, 65536, 65632, 67328, 77568, 141056, 144896, 148992, 153088, 170496, 180224, 219136, 221184, 262144, 281344, 422400, 567296, 569344, 630784, 786432, 831488, 886784, 974848, 1001984, 1048576, 1052672, 1056768, 1464320, 2521088, 3407872, 4382720, 5013504]

We then compute a vectorial addition chain of length 214 from this chain. From this the multi-exponentiation requires $214 - 60 = 154$ multiplications in $\mathbb{F}_{p^{24}}$ including three squarings, where 60 is the number of unit vectors.

Performing the final exponentiation with the vectorial chain requires

$$773M_1 + I_1 + 682S_c + 32M_{24} + 910M_1 + 3S_c + 151M_{24} = 68319M_1 + I_1,$$

which is lower than the cost that we evaluated above. Hence the method with performing the multi-exponentiation is better than the naive method when $k = 24$.

Consequently, the estimated cost of the twisted Ate pairing is $153495M_1 + I_1$.

Weil Pairing with Automorphism

Some of the features that are used to accelerate the Weil pairing described in Section 5.4.3 can also be used when $k = 24$. T_G is $60M_1$ based on pre-computation

with $S_{12} + M_{12}$ and T'_G, which is the cost of evaluating a line function G at the divisor $\psi^{-1}(D_1)$ can be estimated as $402M_1$ with pre-computation with $2(S_{12} + M_{12})$. In each Miller step we compute

$$f^2 \cdot c_D\psi(D'_2) \cdot \overline{c_D(\psi^{-1}(D_1))}$$

with $T_G + T'_G + S_{24} + 2M_{24} = 894M_1$ and

$$f \cdot c_A\psi(D'_2) \cdot \overline{c_A(\psi^{-1}(D_1))}$$

with $T_G + T'_G + M_{24} = 786M_1$.

As described in Section 5.4.3, λ where $\lambda^4 + 1 = 0 \pmod r$, can be written by using t for the Kawazoe-Takahashi cyclotomic family of curves. In fact, the equation

$$(t^3)^4 + 1 = t^{12} + 1 = (t^4 + 1)(t^8 - t^4 + 1) = (t^4 + 1)r(t)$$

implies we can always obtain $\lambda = t^3$ as the smallest integer, which is the root of $y^4 + 1 = 0 \pmod r$. Note that the size of λ is not close to $(\log_2 r)/4$; however, we cannot search for an appropriate λ with a low Hamming weight without incurring excessive computational costs.

Here we have that $\mathbf{D} = 193$ and $\mathbf{A} = 52$ for $\lambda = t^3$. All in all, the cost of the Weil pairing is

$$\begin{aligned} &3(S_{12} + M_{12}) + 193 \cdot (40 + 235 + 894)M_1 + 52 \cdot (41 + 241 + 786)M_1 + M_{24} + I_{24} \\ &= 281860M_1 + I_1. \end{aligned}$$

5.4.5 Comparison

Here we provide a comparison of our pairings on genus 2 curves with the state-of-the art results for pairings on elliptic curves.

Aranha et al. [2] proposed optimal HV pairings on Kachisa-Schaefer-Scott (KSS), Barreto-Naehrig (BN), and Barreto-Lynn-Scott (BLS) elliptic curves at the 192-bit security level. They showed that the pairings on BLS curves were the most efficient and a slight improvement was reported by Ghammam and Fouotsa [42]. They also showed that optimal Weil pairings known as β pairings

were well suited for implementation on a multiprocessor machine and that this achieved an improvement in efficiency.

Zhang et al. [93] analyzed the pairings on KSS curves for the 192-bit security level and on BLS curves for the 256-bit security level. Although their aim is to show the most efficient curves for computing products of pairings at high security levels, they also provide the detailed cost of a single pairing; therefore, we use their results for comparison.

Table 5 and 6 show the comparison of our pairings and other pairings on pairing-friendly elliptic curves at the 192- and 256-bit security levels, respectively.

Table 5. Comparison of the cost of the pairings at the 192-bit security level.

Curve	Pairing	Phase	Operations in $\mathbb{F}_p$
KSS16 [93]	Optimal Ate	ML	$10208m_{481}$
		FE	$22330m_{481} + i_{481}$
		Total	$32538m_{481} + i_{481}$
BLS12 [2]	Optimal Ate	ML	$16775m_{512}$
		FE	$13068m_{512} + i_{512}$
		Total	$29843m_{512} + i_{512}$
Kawazoe-Takahashi16	Twisted Ate	ML	$23601m_{671}$
		FE	$34341m_{671} + i_{671}$
		Total	$57942m_{671} + i_{671}$
Kawazoe-Takahashi16	Variant of Weil	ML	$69218m_{671} + i_{671}$
		FE	—
		Total	$69218m_{671} + i_{671}$

We obtained a comparison of our pairings with those on elliptic curves using scaled data for the size of p by considering the costs in software implementations on 64-bit platforms. As noted in [2, Section 6.2], $a \in \mathbb{F}_p$ can be represented with $\ell = 1 + \lfloor \log_2(p) \rfloor$ binary coefficients a_i packed in $n_{64} = \lceil \frac{\ell}{64} \rceil$ 64-bit processor words. Then the complexity of a Montgomery multiplication is given by $O(2n_{64}^2 + n_{64})$. Hence, we have that $m_{671} \approx 1.86m_{512}$ and $m_{768} \approx 1.43m_{640}$. Based on this, the scaled cost is approximately 3.6 times more than the cost of pairing on the BLS12 curve at the 192-bit security level and approximately 4.4 times more than the cost of pairing on the BLS24 curve at the 256-bit security level, respectively. This is

Table 6. Comparison of the cost of the pairings at the 256-bit security level.

Curve	Pairing	Phase	Operations in $\mathbb{F}_p$
BLS24 [93]	Optimal Ate	ML	$21297m_{640} + 67i_{640}$
		FE	$30596m_{640} + 10i_{640}$
		Total	$51893m_{640} + 77i_{640}$
Kawazoe-Takahashi24	Twisted Ate	ML	$85176m_{768}$
		FE	$68319m_{768} + i_{768}$
		Total	$153495m_{768} + i_{768}$
Kawazoe-Takahashi24	Variant of Weil	ML	$281860m_{768} + i_{768}$
		FE	—
		Total	$281860m_{768} + i_{768}$

attributed to the higher ρ-value of Kawazoe-Takahashi curves in contrast to those of pairing-friendly elliptic curves.

At the 192-bit security level, the cost of Weil pairing with automorphism is competitive with that of twisted Ate pairing, although the cost is much higher at the 256-bit security level. This is a consequence of the loop length of Weil pairing exceeding the optimal length, which is close to $(\log_2 r)/4$. Furthermore, the final exponentiation can be performed with limited effectiveness for $k = 24$ owing to the low cost of cyclotomic squaring in $G_{\Phi_6(p^4)}$.

5.5 Summary

This chapter provided the explicit constructions of the pairing on Kawazoe-Takahashi curves and detailed cost evaluations of them for both 192- and 256-bit security levels. In addition to the Tate-type pairings, the Weil-type pairings with the efficiently computable automorphisms as described in Chapter 4 are evaluated and the cost estimations are reported. At the both two security levels, the twisted Ate pairing is the most efficient Tate-type pairing on the curves.

Two strategies for performing the hard part of final exponentiations are described: for the parametrized coefficients

$$l_i(x) \text{ where } \frac{\Phi_k(p(x))}{r(x)} = \sum_{i=0}^{\varphi(k)} l_i(x)p(x)^i,$$

- **a naive way:** simplifying $l_i(x)$ and computing exponentiations in the efficient order naively,
- **multi-exponentiation:** computing a multi-exponentiation with a vectorial chain [83].

For the 192-bit security levels, the computation of the final exponentiation with naive method is slightly faster than using the method of computing the multi-exponentiation, while computing the multi-exponentiation is better than the naive method for the 256-bit security level.

Based on the cost estimations and the comparison with the results in elliptic case, the scaled cost for a 64-bit processor platform is approximately 3.6 times more than the cost of pairing on the BLS12 curve at the 192-bit security level and approximately 4.4 times more than the cost of pairing on the BLS24 curve at the 256-bit security level, respectively.

6. Implementation

This chapter shows benchmark results of arithmetic in the extension fields and our pairings which are evaluated in chapter 5 on a 64-bit processor platform.

6.1 Modular Multiplications

This section reports a brief overview of algorithms of standard modular arithmetic, especially Montgomery multiplication. The algorithms in this section are typical and the essential parts of them are used many libraries of arithmetic in finite fields which offer optimized codes on each platform.

Representation

For a 64-bit platform, integers are represented in radix 2^{64} using arrays of 64-bit words. For instance, the n_W-word array $(x_0, \ldots, x_{n_W-1})$ represents the $(64n_W)$-bit integer $X = \sum_{i=0}^{n_W-1} x_i \cdot 2^{64i}$.

Basic Integer Operations

Most of the basic arithmetic operations, such as integer addition, subtraction, comparison, assignment, and so on, can be performed where their latencies are linear in n_W. We illustrate this by detailing the case of the addition in the following paragraphs.

Suppose then that we are given the address in memory of two n_W-word integers $X = (x_0, \ldots, x_{n_W-1})$ and $Y = (y_0, \ldots, y_{n_W-1})$, and that we want to compute their sum as the n_W-word integer $R = (r_0, \ldots, r_{n_W-1})$ along with the carry-out bit c: $X + Y = R + c \cdot 2^{64n_W}$.

Basic Modular Arithmetic

Basic modular operations such as negation, addition or subtraction directly rely on their integer counterparts on n_W-word operands described as above. Operands are assumed to be already reduced with respect to the modulus p.

After the main operation, a final reduction step compares the result to the modulus p and conditionally subtracts or adds it (in the case of a modular addi-

tion or subtraction, respectively). This comparison is performed most-significant digits first, so as to return an answer as quickly as possible. Thus, it has an average latency of only a few cycles, even though its worst-case complexity (in the case of equality) is still linear in n_W.

Integer Multiplication

Given two n_W-word multi-precision integers X and Y, their $2n_W$-word product $R = X \cdot Y$ is computed using a quadratic parallel–serial algorithm: the n_W words of the multiplicand X are first all loaded into registers, then, for i ranging from 0 to $n_W - 1$, each partial product $X \cdot y_i$ is computed, shifted left by i words, and accumulated into the partial result:

$$
\begin{aligned}
&R \leftarrow 0 \\
&\textbf{for } i \leftarrow 0 \textbf{ to } n_W - 1 \textbf{ do} \\
&\qquad R \leftarrow R + X \cdot y_i \cdot 2^{64i} \\
&\textbf{return } R
\end{aligned}
$$

Note that each partial product $X \cdot y_i$ fits on $n_W + 1$ words, and that, before the i-th partial product is accumulated, the most-significant words r_{n_W+i} to r_{2n_W-1} of the partial result are all 0. Furthermore, because of the left shift by i words, this means that the accumulation into R will only modify words r_i to r_{n_W+i}, and the carry need not be propagated further. Also, after accumulating the i-th partial product, the i-th word r_i will have reached its final value, and may then be written back to memory. Consequently, at any point in the algorithm, only $n_W + 1$ words of the partial result (from r_i to r_{n_W+i}) need to be kept in the register file. Hence, the total number of registers required for the multiplication is $2n_W + O(1)$.

In order to simplify the carry propagation when accumulating each partial product $X \cdot y_i$ into R, the words x_j of the multiplicand X are processed separately according to the parity of their index j: we write $X = X_0 + X_1 \cdot 2^{64}$, with

$$
\begin{aligned}
X_0 &= \sum_{k=0}^{\lceil n_W/2 \rceil - 1} x_{2k} \cdot 2^{64k}, \qquad \text{and} \\
X_1 &= \sum_{k=0}^{\lfloor n_W/2 \rfloor - 1} x_{2k+1} \cdot 2^{64k}.
\end{aligned}
$$

This way, we first compute the sub-product $S_0^{(i)} = X_0 \cdot y_i$, whose individual products $x_{2k} \cdot y_i \cdot 2^{64k}$ are contiguous but do not overlap, and directly accumulate it into R. We then compute the second sub-product $S_1^{(i)} = X_1 \cdot y_i$, which is also contiguous and overlap-free, and finally accumulate it into R as well.

Montgomery Reduction

Given an odd n_W-word modulus p along with the constant $R = 2^{64n_W}$, the *Montgomery reduction* [72] of a $2n_W$-word integer $X < p \cdot R$ with respect to p is defined as $\mathrm{REDC}_p(X) = X \cdot R^{-1} \bmod p$. As $p < R$, using the *Montgomery representation* of integers modulo p, in which the elements $X \in \mathbb{Z}/p\mathbb{Z}$ are represented by $\widetilde{X} = X \cdot R \bmod p$, the product $Z = X \cdot Y \bmod p$ of two such residues X and $Y \in \mathbb{Z}/p\mathbb{Z}$ can then be computed as $\widetilde{Z} = X \cdot Y \cdot R \bmod p = \mathrm{REDC}_p(\widetilde{X} \cdot \widetilde{Y})$.

Given the precomputed constant $\widetilde{R} = R^2 \bmod p$, conversions to and from this representation can be computed using only n_W-word integer multiplications and Montgomery reductions, as $\widetilde{X} = \mathrm{REDC}_p(X \cdot \widetilde{R})$ and $X = \mathrm{REDC}_p(\widetilde{X})$, respectively.

Finally, as it is also compatible with addition, subtraction and negation modulo p, we can perform all the computations required for ECM in Montgomery representation in order to avoid conversions before and after each modular multiplication.

In [72], Montgomery gives an efficient algorithm requiring only multiplications for computing $\mathrm{REDC}_p(X)$, provided that the 1-word constant $n' = (-p)^{-1} \bmod 2^{64}$ is known (thanks to a precomputation, for instance):

$T \leftarrow (x_0, \ldots, x_{n_W-1})$ *(i.e., $T \leftarrow X \bmod 2^{64n_W}$)*
for $i \leftarrow 0$ **to** $n_W - 1$ **do**
    $q \leftarrow t_0 \cdot n' \bmod 2^{64}$
    $T \leftarrow x_{n_W+i} \cdot 2^{64(n_W-1)} + (T + q \cdot p)/2^{64}$
if $T \geqslant p$ **then**
    $T \leftarrow T - p$
return T

The partial result T is first initialized with the n_W least significant words of X. Then, at each iteration, a multiple of p is added to it so as to make it divisible

by 2^{64}. The value of T is then shifted right by one word, and the next word of X is loaded and added (with carry) to t_{n_W-1}. A single final subtraction of p might be necessary to keep the result below p.

At any point in the algorithm, T is an n_W-word integer along with a delayed carry bit, and thus occupies $n_W + 1$ registers denoted by $\boldsymbol{t}_0$ to $\boldsymbol{t}_{n_W}$. As the n_W-word modulus p is also kept in the register file ($\boldsymbol{n}_0$ to $\boldsymbol{n}_{n_W-1}$), the total number of registers required for this algorithm is then $2n_W + O(1)$.

In fact, this algorithm is in many ways quite similar to that of the parallel–serial multiplication described in the previous part about integer multiplications. In particular, by considering the odd- and even-indexed words of p and by writing $p = p_0 + p_1 \cdot 2^{64}$ as we did for X in the multiplication, we can also split the computation of the partial product $q \cdot p$ into two sub-products $S_0 = q \cdot p_0$ and $S_1 = q \cdot p_1$ and accumulate them separately into T. The only difference is that both accumulations into T might generate output carries.

6.2 Implementation on Haswell CPUs with the Library `mcl`

For generating optimized codes to perform field arithmetic flexibly, the library `mcl` proposed by Mitsunari [69] is suitable. This library is used for computing the pairing on BN curve [71] and the result achieves the state-of-the-art performance of pairings at the 128-bit security level [1]. Note that the pairing in [17, 71] has 126-bit security level for its actual parameters.

6.2.1 The Library `mcl`

The library `mcl` offers a class of a prime field $\mathbb{F}_p$ so that we can take a modulus p and implement standard arithmetic in $\mathbb{F}_p$ flexibly. The codes for generating optimized assembly codes to perform standard arithmetic, i.e., addition, subtraction, Montgomery multiplication, and Montgomery inversion are written carefully. In the latest version, we can take a modulus p whose bit size is less than or equal to 768. The inside of `mcl`, Xbyak [70] which is a x86/x64 just-in-time assembler is used for to generate fast codes. The implementation results in [17] can be obtained by using the former version of `mcl`.

Mitsunari [71] improved the implementation of a multiplication of a 256-bit

integers and a 64-bit integer using `mulx` instruction which is supported by Haswell processors. Since the instruction `mulx` does not affect the carry flag (CF), we don't have to perform carry-preserving additions and can remove some `mov` instructions consequently. Detailed optimizations are described in [71, Section 4].

6.2.2 Multiplications in $\mathbb{F}_{p^2}$

As described in [17, Section 5.2], we can use the method of lazy reduction for performing a multiplication in $\mathbb{F}_{p^2}$ with Montgomery multiplications in the base field. A Montgomery multiplication of a and b in $\mathbb{F}_p$ consists of two parts:

1. Performing the straightforward integer multiplication ab where the bit size of ab is at most twice those of a and b.

2. Performing the Montgomery reduction $\mathrm{REDC}_p(ab)$.

For a multiplication in $\mathbb{F}_{p^2}$, we can reduce the number of the Montgomery reductions by keeping the intermediate results of the double size products and performing additions and subtractions of them. If the latency of a Montgomery reduction is considerable higher than that of a straightforward integer multiplication then the lazy reduction method can work well.

In this research, we adopt the lazy reduction method for a selected modulus by taking benchmark result of multiplication in $\mathbb{F}_{p^2}$ into account.

6.3 Experimental Results

Here, p_{671} and p_{768} denote the 671- and 768-bit modulus p described in 5.4.3 and 5.4.4, respectively. For multiplications and squarings in $\mathbb{F}_{p^2}$, both methods of naive way and lazy reduction are considered and implemented. Table 7 shows latencies of a multiplication and a squaring in the quadratic extension fields $\mathbb{F}_{p^2}$ for the naive method and the lazy reduction method.

The benchmark results indicated that a multiplication with lazy reduction in $\mathbb{F}_{p_{671}^2}$ is more effective than a multiplication with naive method. Furthermore we should perform a squaring in $\mathbb{F}_{p_{671}^2}$, and a multiplication and a squaring in $\mathbb{F}_{p_{768}^2}$ in the naive way. Hence only multiplications in $\mathbb{F}_{p_{671}^4}, \mathbb{F}_{p_{671}^8}$, and $\mathbb{F}_{p_{671}^{16}}$ are implemented based on a multiplication in $\mathbb{F}_{p_{671}^2}$ with lazy reduction.

Table 7. Latencies of a multiplication and a squaring in $\mathbb{F}_{p^2}$.

Operation	over $\mathbb{F}_{p_{671}^2}$	over $\mathbb{F}_{p_{768}^2}$
Multiplication (naive)	2,324	2,723
Multiplication (lazy reduction)	2,194	2,749
Squaring (naive)	1,582	2,007
Squaring (lazy reduction)	1,774	2,345

The following tables 8 and 9 report the latency of standard arithmetic in base field and each extension field for p_{671} and p_{768}, respectively.

Table 8. Latency of arithmetic in each field of characteristic p_{671}

Operation	Latency
Addition over $\mathbb{F}_{p_{671}}$	25
Inversion over $\mathbb{F}_{p_{671}}$	9,468
Cyclotomic squaring over $G_{\Phi_2(p^8)}$	37,158

Field	Multiplication	Squaring
$\mathbb{F}_{p_{671}}$	720	713
$\mathbb{F}_{p_{671}^2}$	2,194	1,582
$\mathbb{F}_{p_{671}^4}$	7,057	5,125
$\mathbb{F}_{p_{671}^8}$	21,545	15,741
$\mathbb{F}_{p_{671}^{16}}$	65,464	48,448

Table 10 shows benchmark results for the pairings on Kawazoe-Takahashi curves on the Haswell CPU, Intel Core i7-4770K. To perform arithmetic in $\mathbb{F}_p$ for 768-bit modulus p, the option `CFLAGS_USER=''-DMCL_MAX_OP_BIT_SIZE=768''` is required. In addition, we use the latest version of LLVM (version 4.0.0) for generating optimal code and set the option `LLVM_VER=-3.8 USE_LLVM=1`.

Aranha et al. [2] provided the experimental results for the optimal Ate pairing on the elliptic curves that are the state-of-the-art for pairings at the 192-bit security level in elliptic case. They implemented field arithmetic in Assembly on top of [3].

For comparison, the benchmark results of pairings for the 192- and 256-bit security levels by Bos et al. [23] are reported. The implementation result of a

Table 9. Latency of arithmetic in each field of characteristic p_{768}

Operation	Latency
Addition over $\mathbb{F}_{p_{768}}$	33
Inversion over $\mathbb{F}_{p_{768}}$	10,794
Cyclotomic squaring over $G_{\Phi_6(p^4)}$	60,080

Field	Multiplication	Squaring
$\mathbb{F}_{p_{768}}$	833	824
$\mathbb{F}_{p_{768}^2}$	2,723	2,007
$\mathbb{F}_{p_{768}^4}$	8,652	6,149
$\mathbb{F}_{p_{768}^{12}}$	53,094	39,508
$\mathbb{F}_{p_{768}^{24}}$	163,341	117,748

pairing for the 256-bit security level can be found in [81]. By Estimating the latency of the pairing with the timing of 'Decrypt' in [81, Table 3], his result should be better than that of the pairing on BLS24 curves in [23].

The latency of the twisted Ate pairing on Kawazoe-Takahashi curves for the 192-bit security level is approximately 4 times higher than that of the optimal Ate pairing on BLS12 curves by [2].

6.4 Summary

This chapter offered the benchmark results of the twisted Ate pairing on the Kawazoe-Takahashi curves. The implementation is performed by using the library `mcl` for arithmetic in base prime field $\mathbb{F}_p$. For the Montgomery multiplications in $\mathbb{F}_{p^2}$, the naive way or the lazy reduction method is adopted depending on the modulus p.

The latency of the twisted Ate pairing on Kawazoe-Takahashi curves for the 192-bit security level is approximately 4 times higher than that of the optimal Ate pairing on BLS12 curves by [2] which is the state-of-the-art result of pairing computation for the security level.

Table 10. Benchmark results for our pairings and a comparison with the pairings on the elliptic curves

Curve and pairing	Platform	Millions of clock cycles
BLS12 [2] optimal Ate	Intel Core i7-2630QM	14.08
BLS12 [23] optimal Ate	Intel Core i7-3520M	47.2
KSS18 [23] optimal Ate	Intel Core i7-3520M	63.3
Kawazoe-Takahashi16 twisted Ate	Intel Core i7-4770K	52.42
BLS24 [23] optimal Ate	Intel Core i7-3520M	115.0
Kawazoe-Takahashi24 twisted Ate	Intel Core i7-4770K	153.19

7. Conclusion

Pairing-based cryptography has attracted attention since there are many useful cryptographic protocols with pairings.

In the literature, pairings on hyperelliptic curves of genus 2 are not considered to be more efficient than elliptic curves for constructing general pairings. In fact we already have optimum families of curves for effective pairings in the elliptic case. In contrast to the elliptic case, the general constructions of pairing-friendly genus 2 curves with $\rho < 2$ have not yet been achieved. Furthermore the η_T pairing which was the most efficient pairing on genus 2 curves has been vulnerable due to the major progress in the computation of discrete logarithms in finite fields of small characteristics was made. It is therefore unclear effective constructions of a pairing and how to select an appropriate curve for a fast pairing.

This dissertation explores how to construct the most efficient pairing on genus 2 hyperelliptic curve for the 192- and 256-bit security level. By the detailed cost evaluations and the implementation results of our pairings, the extent of the difference between pairings on elliptic curves and pairings on genus 2 hyperelliptic curves at the security levels was clarified.

7.1 Contributions

To achieve the above objectives, a selection of appropriate curves was revealed in Chapter 3. The Kawazoe-Takahashi families of curves are an optimum choice for an effective pairing at the 192- and 256-bit security levels since they have twists of higher degree and many features to speed up the pairings.

Chapter 4 proposed a new invariant of Weil-type pairing with an efficiently computable automorphism of a curve. This result can be considered as a generalization of the methods that were used to construct the *omega pairing* on the elliptic curve by Zhao et al. [95]. Furthermore, this approach enables Weil-type pairings with automorphisms in the Hess-Vercauteren framework. An effective method for computing the pairing evaluated effective parts of reduced divisors as inputs without normalizing the Miller function was provided.

Chapter 5 reported a detailed cost estimation of Kawazoe-Takahashi curves at the 192- and 256-bits security levels. Constructions of the pairings were analyzed

carefully, and one can see that the twisted Ate pairing can be computed more effectively. For the twisted Ate pairing, the scaled cost is approximately 3.6 and 4.4 times more than the cost of pairing on the BLS12 curve at the 192-bit security level and on the BLS24 curve at the 256-bit security level, respectively. In addition, we reported the cost of our Weil-type pairings with efficiently computable automorphisms.

Chapter 6 offered experimental results for implementation of pairings on a Haswell processor with `mcl` library. Thanks to `mcl` which can generate optimum codes for field arithmetic and implementing multiplications in the first quadratic extension field in the lazy reduction way, the latency of our pairing is approximately 4 higher than the state-of-the-art in elliptic case for the 192-bit security level.

This is the first comprehensive study of the detailed evaluations of pairings on genus 2 curves at the high security levels.

7.2 Future Perspective

There is a considerable difference between cost of pairings on elliptic curves and that of pairings on genus 2 curves. One of the causes comes from the discrepancy of ρ-values. The ρ-values of optimum families of elliptic curves take approximately 1 while those of Kawazoe-Takahashi families in this dissertation take $3 \leqslant \rho \leqslant 3.5$. This is the one of biggest disadvantage for a pairing on genus 2 curves. Therefore, the method for constructing pairing-friendly genus 2 curves with $\rho < 2$ should be exploited.

This dissertation has focused on the cyclotomic *families* by Kawazoe and Takahashi to parametrize security parameters and construct a family of pairings in the Hess-Vercauteren framework. However, the length of Miller loop of the Weil pairing with the automorphism (i.e., λ which is one of roots of the characteristic polynomial of the automorphism) is not desired one since there is a certain relationship between the security parameters $r(x)$ and $p(x)$ for the 256-bit security level. For evaluating pairings on genus 2 curves completely, it needs to construct a pairing on a pairing-friendly curve which is not member of any family. Especially, the Weil-type pairing should be more efficient for higher security levels since no final exponentiation is required to perform. Hence we should

propose efficient Weil-type pairings on appropriate curves in the future.

As described in Section 3.1, we should reconsider appropriate key length of pairings in finite fields $\mathbb{F}_{q^k}$ due to the progress of research on the tower number field sieve (TNFS and exTNFS) algorithms and those methods for a special characteristic p (sTNFS and sexTNFS). In elliptic and genus 2 cases, the pairings on the families of pairing-friendly curves are affected by these algorithms and their key sizes must be taken into large values. The sizes of $\mathbb{F}_{q^k}$ in our pairings are somewhat larger than the current recommended key sizes for the 192- and 256-bit security levels that is caused by their slightly larger ρ-values. In contrast to the genus 2 case, the pairings on the elliptic curves in the literatures should be reconstructed soon.

References

[1] D. F. Aranha, P. S. L. M. Barreto, P. Longa, and J. E. Ricardini. The realm of the pairings. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013: 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 3–25. Springer Berlin Heidelberg, 2014.

[2] D. F. Aranha, L. Fuentes-Castañeda, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography – Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195. Springer Berlin Heidelberg, 2013.

[3] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[4] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 48–68. Springer Berlin Heidelberg, 2011.

[5] D. F. Aranha, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez. Parallelizing the weil and tate pairings. In L. Chen, editor, *Cryptography and Coding: 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, pages 275–295. Springer Berlin Heidelberg, 2011.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, Feb. 2006.

[7] J. Balakrishnan, J. Belding, S. Chisholm, K. Eisenträger, K. E. Stange, and E. Teske. Pairings on hyperelliptic curves. CoRR, abs/0908.3731, Available: `http://arxiv.org/abs/0908.3731v2`, 2009.

[8] R. Barbulescu, P. Gaudry, A. Guillevic, and F. Morain. Improving NFS for the discrete logarithm problem in non-prime finite fields. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 129–155. Springer Berlin Heidelberg, 2015.

[9] R. Barbulescu, P. Gaudry, A. Guillevic, and F. Morain. Improving nfs for the discrete logarithm problem in non-prime finite fields. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 129–155. Springer Berlin Heidelberg, 2015.

[10] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In P. Nguyen and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2014.

[11] R. Barbulescu, P. Gaudry, and T. Kleinjung. The tower number field sieve. In T. Iwata and H. J. Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 – December 3, 2015, Proceedings, Part II*, pages 31–55. Springer Berlin Heidelberg, 2015.

[12] P. S. L. M. Barreto, S. D. Galbraith, C. Ó. hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.

[13] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings*, pages 354–369. Springer Berlin Heidelberg, 2002.

[14] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, G. Persiano, and C. Galdi, editors, *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers*, pages 257–267. Springer Berlin Heidelberg, 2003.

[15] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 319–331. Springer Berlin Heidelberg, 2006.

[16] N. Benger and M. Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In M. A. Hasan and T. Helleseth, editors, *Arithmetic of Finite Fields: Third International Workshop, WAIFI 2010, Istanbul, Turkey, June 27-30, 2010. Proceedings*, pages 180–195. Springer Berlin Heidelberg, 2010.

[17] J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing-Based Cryptography - Pairing 2010: 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, pages 21–39. Springer Berlin Heidelberg, 2010.

[18] BlueKrypt. - cryptographic key length recommendation, `http://www.keylength.com`, 2012.

[19] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 213–229. Springer Berlin Heidelberg, 2001.

[20] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances*

in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings, pages 258–275. Springer Berlin Heidelberg, 2005.

[21] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[22] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 253–273. Springer Berlin Heidelberg, 2011.

[23] J. W. Bos, C. Costello, and M. Naehrig. Exponentiating in pairing groups. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013: 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 438–455. Springer Berlin Heidelberg, 2014.

[24] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[25] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Designs, Codes and Cryptography*, 37(1):133–141, 2005.

[26] Y. Choie and E. Lee. Implementation of tate pairing on hyperelliptic curves of genus 2. In J.-I. Lim and D.-H. Lee, editors, *Information Security and Cryptology - ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers*, pages 97–111. Springer Berlin Heidelberg, 2004.

[27] C. Cocks and R. Pinch. Identity-based cryptosystems based on the weil pairing, 2001. unpublished manuscript.

[28] A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over barreto-naehrig curves. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007: First*

International Conference, Tokyo, Japan, July 2-4, 2007. Proceedings, pages 197–207. Springer Berlin Heidelberg, 2007.

[29] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, Sept. 2006.

[30] B. Dixon and A. Lenstra. Massively parallel elliptic curve factoring. In R. Rueppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 183–193. Springer Berlin Heidelberg, 1993.

[31] J. Fan, F. Vercauteren, and I. Verbauwhede. Faster $\mathbb{F}_p$-arithmetic for cryptographic pairings on barreto-naehrig curves. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, pages 240–253. Springer Berlin Heidelberg, 2009.

[32] X. Fan, G. Gong, and D. Jao. Speeding up pairing computations on genus 2 hyperelliptic curves with efficiently computable automorphisms. In S. Galbraith and K. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 243–264. Springer Berlin Heidelberg, 2008.

[33] X. Fan, G. Gong, and D. Jao. Efficient pairing computation on genus 2 curves in projective coordinates. In R. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2009.

[34] D. Freeman. Constructing pairing-friendly genus 2 curves with ordinary jacobians. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007: First International Conference, Tokyo, Japan, July 2-4, 2007. Proceedings*, pages 152–176. Springer Berlin Heidelberg, 2007.

[35] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[36] D. Freeman, P. Stevenhagen, and M. Streng. Abelian varieties with prescribed embedding degree. In A. J. van der Poorten and A. Stein, editors, *Algorithmic Number Theory: 8th International Symposium, ANTS-VIII Banff, Canada, May 17-22, 2008 Proceedings*, pages 60–73. Springer Berlin Heidelberg, 2008.

[37] D. M. Freeman and T. Satoh. Constructing pairing-friendly hyperelliptic curves using weil restriction. *Journal of Number Theory*, 131(5):959 – 983, 2011. Elliptic Curve Cryptography.

[38] S. D. Galbraith, F. Hess, and F. Vercauteren. Hyperelliptic pairings. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 108–131. Springer Berlin Heidelberg, 2007.

[39] S. D. Galbraith, X. Lin, and D. J. M. Morales. Pairings on hyperelliptic curves with a real model. In S. Galbraith and K. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 265–281. Springer-Verlag, 2008.

[40] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, sep 2008.

[41] S. D. Galbraith, J. Pujolàs, C. Ritzenthaler, and B. Smith. Distortion maps for supersingular genus two curves. *Journal of Mathematical Cryptology*, 3(1):1–18, 2009.

[42] L. Ghammam and E. Fouotsa. On the computation of the optimal ate pairing at the 192-bit security level. Cryptology ePrint Archive, Report 2016/130, 2016. `http://eprint.iacr.org/2016/130`.

[43] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98. ACM, 2006.

[44] R. Granger, F. Hess, R. Oyono, N. Thériault, and F. Vercauteren. Ate pairing on hyperelliptic curves. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings*, pages 430–447. Springer Berlin Heidelberg, 2007.

[45] R. Granger, D. Page, and N. P. Smart. High security pairing-based cryptography revisited. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory: 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006. Proceedings*, pages 480–494. Springer Berlin Heidelberg, 2006.

[46] R. Granger and M. Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography – PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 209–223. Springer Berlin Heidelberg, 2010.

[47] A. Guillevic and D. Vergnaud. Genus 2 hyperelliptic curve families with explicit jacobian order evaluation and pairing-friendly constructions. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography – Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 234–253. Springer Berlin Heidelberg, 2013.

[48] D. Hankerson, A. Menezes, and M. Scott. Software implementation of pairings. In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, pages 188–206. IOS Press, 2008.

[49] F. Hess. Pairing lattices. In S. Galbraith and K. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 18–38. Springer-Verlag, 2008.

[50] F. Hess, N. P. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Trans. Information Theory*, 52(10):4595–4602, 2006.

[51] J. Jeong and T. Kim. Extended tower number field sieve with application

to finite fields of arbitrary composite extension degree. Cryptology ePrint Archive, Report 2016/526, 2016. http://eprint.iacr.org/2016/526.

[52] A. Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

[53] A. Joux and C. Pierrot. The special number field sieve in $\mathbb{F}_{p^n}$, application to pairing-friendly constructions. In Z. Cao and F. Zhang, editors, *Pairing-Based Cryptography – Pairing 2013: 6th International Conference, Beijing, China, November 22-24, 2013, Revised Selected Papers*, pages 45–61. Springer International Publishing, 2014.

[54] E. Kachisa. Generating more kawazoe-takahashi genus 2 pairing-friendly hyperelliptic curves. In M. Joye, A. Miyaji, and A. Otsuka, editors, *Pairing-Based Cryptography – Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 312–326. Springer Berlin Heidelberg, 2010.

[55] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008: Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, pages 126–135. Springer Berlin Heidelberg, 2008.

[56] D. Kammler, D. Zhang, P. Schwabe, H. Scharwaechter, M. Langenberg, D. Auras, G. Ascheid, and R. Mathar. Designing an asip for cryptographic pairings over barreto-naehrig curves. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, pages 254–271. Springer Berlin Heidelberg, 2009.

[57] K. Karabina. Squaring in cyclotomic subgroups. *Math. Comput.*, 82(281), 2013.

[58] M. Kawazoe and T. Takahashi. Pairing-friendly hyperelliptic curves with ordinary jacobians of type $y^2 = x^5 + ax$. In S. Galbraith and K. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 164–177. Springer Berlin Heidelberg, 2008.

[59] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 543–571. Springer Berlin Heidelberg, 2016.

[60] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

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

[62] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. P. Smart, editor, *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings*, pages 13–36. Springer Berlin Heidelberg, 2005.

[63] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, 2005.

[64] E. Lee, H. Lee, and C. Park. Efficient and generalized pairing computation on abelian varieties. *IACR Cryptology ePrint Archive*, 2008:40, 2008.

[65] M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts (special section on cryptography and information security). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(1):54–63, jan 1997.

[66] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[67] S. V. Miller. The weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[68] V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426. Springer Berlin Heidelberg, 1986.

[69] S. Mitsunari. mcl, a class library, https://github.com/herumi/mcl.

[70] S. Mitsunari. Xbyak, a JIT assembler library, https://github.com/herumi/xbyak.

[71] S. Mitsunari. A fast implementation of the optimal ate pairing over BN curve on intel haswell processor. *IACR Cryptology ePrint Archive*, 2013:362, 2013.

[72] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[73] D. Mumford. *Abelian varieties*. Tata Institute of fundamental research studies in mathematics. Oxford University Press, 1974.

[74] C. Ó hÉigeartaigh and M. Scott. Pairing calculation on supersingular genus 2 curves. In E. Biham and A. M. Youssef, editors, *Selected Areas in Cryptography: 13th International Workshop, SAC 2006*, volume 4356 of *Lecture Notes in Computer Science*, pages 302–316. Springer Berlin Heidelberg, 2007.

[75] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[76] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, pages 457–473. Springer Berlin Heidelberg, 2005.

[77] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairings. In *Symposium on Cryptography and Information Security — SCIS 2000*, Japan, 2000.

[78] M. Scott. Faster pairings using an elliptic curve with an efficient endomorphism. In S. Maitra, C. E. Veni Madhavan, and R. Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005: 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005. Proceedings*, pages 258–269. Springer Berlin Heidelberg, 2005.

[79] M. Scott. Scaling security in pairing-based protocols. Cryptology ePrint Archive, Report 2005/139, 2005. `http://eprint.iacr.org/2005/139`.

[80] M. Scott. Implementing cryptographic pairings. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 177–196. Springer Berlin Heidelberg, 2007.

[81] M. Scott. On the efficient implementation of pairing-based protocols. In *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, pages 296–308, 2011.

[82] M. Scott and P. S. L. M. Barreto. Compressed pairings. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings*, pages 140–156. Springer Berlin Heidelberg, 2004.

[83] M. Scott, N. Benger, M. Charlemagne, L. Dominguez Perez, and E. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88. Springer Berlin Heidelberg, 2009.

[84] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate texts in mathematics*. Springer, 1986.

[85] J. H. Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*, volume 151 of *Graduate Texts in Mathematics*. Springer, 1994.

[86] P. Smith and C. Skinner. A public-key cryptosystem and a digital signature system based on the lucas function analogue to discrete logarithms.

In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology — ASIACRYPT'94: 4th International Conferences on the Theory and Applications of Cryptology Wollongong, Australia, November 28 – December 1, 1994 Proceedings*, pages 355–364. Springer Berlin Heidelberg, 1995.

[87] M. Stam and A. K. Lenstra. Speeding up xtr. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*, pages 125–143. Springer Berlin Heidelberg, 2001.

[88] M. Stam and A. K. Lenstra. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In B. S. Kaliski, ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers*, pages 318–332. Springer Berlin Heidelberg, 2003.

[89] T. Teruya, K. Saito, N. Kanayama, Y. Kawahara, T. Kobayashi, and E. Okamoto. Constructing symmetric pairings over supersingular elliptic curves with embedding degree three. In Z. Cao and F. Zhang, editors, *Pairing-Based Cryptography – Pairing 2013*, volume 8365 of *Lecture Notes in Computer Science*, pages 97–112. Springer-Verlag, 2014.

[90] F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

[91] E. R. Verheul. Self-blindable credential certificates from the weil pairing. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*, pages 533–551. Springer Berlin Heidelberg, 2001.

[92] F. Zhang. Twisted ate pairing on hyperelliptic curves and applications. *Science China Information Sciences*, 53(8):1528–1538, 2010.

[93] X. Zhang and D. Lin. Analysis of optimum pairing products at high security levels. In S. Galbraith and M. Nandi, editors, *Progress in Cryptology -*

INDOCRYPT 2012: 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings, pages 412–430. Springer Berlin Heidelberg, 2012.

[94] X. Zhang and K. Wang. Fast symmetric pairing revisited. In Z. Cao and F. Zhang, editors, *Pairing-Based Cryptography – Pairing 2013*, volume 8365 of *Lecture Notes in Computer Science*, pages 131–148. Springer-Verlag, 2014.

[95] C.-A. Zhao, D. Xie, F. Zhang, J. Zhang, and B.-L. Chen. Computing bilinear pairings on elliptic curves with automorphisms. *Designs, Codes and Cryptography*, 58(1):35–44, 2011.

Achievements

Journal

- M. Ishii, A. Inomata and K. Fujikawa. A Weil Pairing on a Family of Genus 2 Hyperelliptic Curves with Efficiently Computable Automorphisms, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Special Section on Cryptography and Information Security*, 2017 (to be appeared).

International Conference

- M. Ishii, A. Inomata and K. Fujikawa. A Construction of a Twisted Ate Pairing on a Family of Kawazoe–Takahashi Curves at 192-bit Security Level and Its Cost Estimate, In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pp. 432–439, 2016.

- M. Ishii, A. Inomata and K. Fujikawa. Parallel GPU Implementation of η_T Pairing over Fields of Characteristic Two, In *ICNCS 2014, International Journal of Computer and Communication Engineering*, Vol. 3, No. 3, pp. 193-198, 2014.