Implementing Cryptography Pairings over Ordinary Pairing-Friendly Curves of Type $y^2 = x^5 + a\,x$

Mohammed Zitouni(✉) and Farid Mokrane

Laboratory Analysis, Geometry and Applications CNRS (UMR 7539), Galilee Institute, Paris 13 University, Villetaneuse, France
zitounimohammmed@gmail.com, mokrane@math.univ-paris13.fr

Abstract. In this paper, we describe an efficient implementation in Sage of the Tate pairing over ordinary hyperelliptic curves of type $y^2 = x^5 + a\,x$. First, we describe a method of construction of these curves according to Kawazoe and Takahashi [8]. Then, we describe an efficient formula for computing pairings on such curves over prime fields, and develop algorithms to compute Tate pairing. We provide a faster optimisation of the final exponentiation in particular for the embedding degree $k = 28$.

Keywords: Hyperelliptic curve · Tate pairing · Finite field · Embedding degree · Final exponentiation

1 Introduction

In 1989, three years after the introduction of elliptic curves cryptography, Koblitz suggest to use hyper-elliptic curves as a generalization to higher genus curves [8]. He extended the idea of abelian points group on elliptic curves over finite field to the Jacobian of a hyperelliptic curves, since the Jacobian is a finite abelian group on which the arithmetic operations are applied.

The algebraic curves-based cryptography has divided the cryptographers community [11] into two teams. The first one which argues that the problems of factorization and discrete logarithm problem (DLP) over finite field have already been intensively studied; and that it would require more time before the community can really apprehend the nature of elliptic curves. The other team was ambitious, started working in it and proposed their first protocols.

The first protocols proposed are based on a mathematical tool called pairing, the oldest of them being the Weil pairing. This mathematical protocol has received a great attention by the researchers and is now among the majors topics in cryptography. In order to realize protocols based on pairings, it is essential to have Pairing-friendly-curves which have parameters such as p the large prime fields $\mathbb{F}_p$ and the embedding degree k. The embedding degree plays an important role in ensuring a certain desired level security. In this context, the security of the pairing-based cryptosystems depends on finding curves whose Jacobian order

E. Simion and R. Géraud-Stewart (Eds.): SecITC 2019, LNCS 12001, pp. 95–108, 2020.
https://doi.org/10.1007/978-3-030-41025-4_7

over the finite fields $\mathbb{F}_{p^k}$, is divisible by a larger prime number ℓ (a condition necessary for resisting attacks such as Pohling-Hellman attacks).

In this paper, we consider Kawazoe-Takahashi [8] genus two ordinary pairing-friendly curves of type $y^2 = x^5 + a\,x$ which are generated over a finite field $\mathbb{F}_{p^k}$ using a method introduced by Kachisa [7]. These curves are characterized by simple and fast complex multiplications. It is also possible to have curves with a known odd prime factor ℓ of the Jacobian order with different embedding degrees and offer a small $\rho_{-value} = 2\,log(p)/log(\ell)$ between 2 and 3.

The paper is organized as follows. First, we recall some backgrounds on pairings over hyperelliptic curves, Jacobian group structure, and representation of divisors classes. We describe a method to construct ordinary pairing-friendly of Kawazoe and Takahashi curves with simple method proposed by Kachisa [7] to obtain curves with small ρ_{-value}. Then, we recall the Tate-Lichtenbaum pairing definition and present implementation techniques for pairings on hyperelliptic curves. We provide a new approach to the final exponentiation when the embedding degree is $k = 28$. The critical computational task of evaluating a function at a divisor is also provided.

Finally, we give some implementation results in Sage using Intel Core i5-7300HQ CPU @ 2.50 GHz processor on several security levels. We conclude that for most applications there exits an efficient algorithm for computing pairings on hyperelliptic curves that is better that on ordinary elliptic curves from the point of view of efficiency and security.

2 Preliminaries

In this section, we briefly recall the definition of the hyperelliptic curves, pairings and the definition of the Tate-Lichtenbaum pairing.

2.1 Hyperelliptic Curves

A genus g hyperelliptic curves over a prime finite field $\mathbb{F}_p$ are non-singular curves of a general form:

$$\mathcal{H}: \qquad y^2 + h(x)\,y = f(x) \tag{1}$$

with $h,\ f \in \mathbb{F}_p[x]$, $deg(f) = 2\ g + 1$, $deg(h) \leq g$ and $f(x)$ is monic. For any algebraic extension $\mathbb{K}$ of $\mathbb{F}_p$, there is a special point at infinity, which is denoted by P_∞, and we can consider the set of K-rational points on $\mathcal{H}$:

$$\mathcal{H}(K) := \{(x, y) \in K \times K \mid y^2 + h(x)\,y - f(x) = 0\} \cup \{\infty\}. \tag{2}$$

2.2 Pairings

The concept of a pairing was introduced in cryptography for the fitst time by Menezes et al. [11] to attack instances of the Discrete Logarithm Problem on

elliptic curves and hyperelliptic curves. In 2000, pairing was used as bilinear application by Joux [6] to build cryptographic protocols.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ two additive abelian groups of prime order ℓ, $\mathbb{G}_3$ a multiplicative abelian group of order also ℓ. A bilinear pairing on $(\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_3)$ is a map:

$$e : (\mathbb{G}_1, +)\ \times\ (\mathbb{G}_2, +) \quad \longrightarrow \quad (\mathbb{G}_3, \times)$$

that satisfies the following requirements:

1. Bilinearity : $\forall D_1, D_1^{'} \in \mathbb{G}_1, \forall D_2, D_2^{'} \in \mathbb{G}_2$,
   i. $e(D_1 + D_1^{'}, D_2) = e(D_1, D_2)\ e(D_1^{'}, D_2)$,
   ii. $e(D_1, D_2 + D_2^{'}) = e(D_1, D_2)\ e(D_1, D_2^{'})$,
   iii. $e(a\ D_1, D_2) = e(D_1, a\ D_2) = e(D_1, D_2)^a, a \in \mathbb{N}^*$.
2. Non-degeneracy:
   i. $\forall D_1 \in \mathbb{G}_1 - \{0\},\ \exists D_2 \in \mathbb{G}_2\ :\ e(D_1, D_2)\ \neq\ 1$,
   ii. $\forall D_2 \in \mathbb{G}_2 - \{0\},\ \exists D_1 \in \mathbb{G}_1\ :\ e(D_1, D_2)\ \neq\ 1$.
3. Easily and efficiently calculable.

We recall here the definition of the Tate-Lichtenbaum pairing as it is stated in the literature, which is an explicit version described by Lichtenbaum.

Let $Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$ be the Jacobian group of the hyperelliptic curve $\mathcal{H}$ over $\mathbb{F}_{p^k}$, ℓ be a prime with $\ell \mid \sharp Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$ and let k be the smallest integer such that $\ell \mid (p^k - 1)$, then k is called the embedding degree (dependent on ℓ).

Definition 1. *The Tate-Lichtenbaum pairing is a bilinear and non-degeneracy map defined by:*

$$\begin{aligned} T_\ell :\quad Jac_{\mathcal{H}}(\mathbb{F}_{p^k})[\ell] \times Jac_{\mathcal{H}}(\mathbb{F}_{p^k})/\ell\ Jac_{\mathcal{H}}(\mathbb{F}_{p^k}) \quad &\longrightarrow \quad \mathbb{F}_{p^k}^{\times}/(\mathbb{F}_{p^k}^{\times})^{\ell} \\ (D_1, D_2) \quad &\longmapsto \quad T_\ell(D_1, D_2) = f_{\ell,D_1}(D_2)^{(p^k-1)/\ell}. \end{aligned}$$

f_{ℓ,D_1}: the function given by the divisor $\ell\ D_1\ -\ \ell\ (\infty) = div(f)$.

3 Pairing-Friendly Curves of Type $y^2 = x^5 + a\,x$

3.1 Curve Choice

We can give here explicit constructions of pairing-friendly hyperelliptic curves with ordinary Jacobian proposed by Kawazoe and Takahishi [8], we show also such that curves are suitable to construct genus 2 pairing at the high security levels.

We consider p an odd prime number, $\mathbb{F}_p$ is a finite field of characteristic $p \neq 2$, so we can define equation of the curve $\mathcal{H}$ as $y^2 = f(x)$ where $f(x)$ is a polynomial $\in \mathbb{F}_p[x]$ of degree 5. Let Jac_H be the Jacobian variety of a hyperelliptic curve $\mathcal{H}$. We denote the group of rational points on Jac_H over $\mathbb{F}_p$ by $Jac_{\mathcal{H}}(\mathbb{F}_p)$.

To compute the Jacobian order, we need the characteristic polynomial of $p-$th power Frobenius endomorphism of $\mathcal{H}$. Then the order is given by

$$\sharp\, Jac_{\mathcal{H}}(\mathbb{F}_p) = \chi_p(1)$$

It is very difficult to evaluate the characteristic polynomial of $p-$th power Frobenius endomorphism of $\mathcal{H}$ in 1 for hyperelliptic curves over height level bits fields, there are very few results on it, Gaudry and Harley [3] compute the group order over 80-bits fields but their algorithm needs very long running time. To solve this problem, there are a very special curves with complex multiplication, they are known by the existence of efficient algorithms called CM-methods to construct such curves. The best example of such ordinary pairing-friendly curves is curves of type $y^2 = x^5 + a\,x$ given by Kawazoe and Takahashi [8]. They proposed also a fast algorithm to compute the Jacobian group order over a prime finite field.

3.2 Counting Points

In [3] Gaudry presented a method to compute the Jacobian order modulo the characteristic p of the base field by using the Hasse-Witt matrix. Two main theorems in [10] and [16] quoted below:

Theorem 1. *Let $y^2 = f(x)$ with $deg(f) = 2g + 1$ be the equation of a genus g hyperelliptic curve. Denote by c_i the coefficient of x^i in the polynomial $f(x)^{(p-1)/2}$. Then the Hasse-Witt matrix is given by*

$$A = (c_{ip-j})_{1\leq i,j\leq g}.$$

The following theorem give the link between the characteristic polynomial of the Frobenius endomorphism and the Hasse-Witt matrix.

Theorem 2. *Let $\mathcal{H}$ be a curve of genus g defined over a finite field $\mathbb{F}_{p^k}$. Let A be the Hasse-Witt matrix of $\mathcal{H}$, and let $A_\phi = AA^{(p)}...A^{(p^{k-1})}$. Let $\kappa(t)$ be the characteristic polynomial of the matrix A_ϕ and $\chi(t)$ the characteristic polynomial of the Frobenius endomorphism. Then*

$$\chi(t) \equiv (-1)^g\, t^g\, \kappa(t)(mod\ p).$$

This method is difficult in general when p is very large, but if we consider a special form of $f(x) = x^5 + a\,x \in \mathbb{F}_p[x]$ of degree 5 we can easily compute the Hasse-Witt matrix, $A = \begin{bmatrix} c_{p-1} & c_{p-2} \\ c_{2p-1} & c_{2p-2} \end{bmatrix}$, the element (c_i) is coefficient of x^i in polynomial $f(x)^{(p-1)/2}$. The characteristic polynomial of the Frobenius endomorphism of the genus 2 curve $y^2 = f(x)$ over $\mathbb{F}_p$ is

$$\chi(t) = t^4 - s_1\, t^3 + s_2\, t^2 - s_1\, p\, t + p^2, \qquad |s_1| \leq 4\sqrt{p}, \quad |s_2| \leq 6\, p$$

The s_1 and s_2 are two integers (for more details see Theorem 3, [3]), they are given by

$$s_1 \equiv c_{p-1} + c_{2p-2}(mod\ p) \quad and \quad s_2 \equiv c_{p-1}\, c_{2p-2} + c_{p-2}\, c_{2p-1}(mod\ p)$$

3.3 Pairing-Friendly Curves of the Cocks-Pinch Method

By using results in [Theorem 3, [3]] and Cocks-Pinch method for hyperelliptic curve $\mathcal{H} : y^2 = x^5 + a\,x$. We can construct pairing-friendly curves of this type over a prime field $\mathbb{F}_p$, with parameters c and d integers such that: $p = c^2 + 2\,d^2$, ℓ large prime factor of the Jacobian order over $\mathbb{F}_p$ and k embedding degree, satisfying the following conditions:

1. $p \equiv 1,\ 3 \pmod 8$,
2. $\chi \equiv 0 \pmod \ell$,
3. $\phi_k(p) \equiv 0 \pmod \ell$,
4. $p = c^2 + 2\,d^2, \quad with \quad c \equiv 1 \pmod 4$.

We did the implementation in Sage of the algorithm presented by Kawazoe and Takahashi which is the analog of Cocks-Pinch method to obtain genus 2 ordinary hyperelliptic curves of the form $y^2 = x^5 + a\,x$. By using generalization of Kachisa [7] which he parametrized the parameters c, d, r and p as polynomials $c(z)$, $d(z)$, $\ell(z)$ and $p(z)$ in a variable z. By using this approach, we can obtain curves with small $\rho - value = 2\,log(p)/log(\ell)$.

Algorithm 1. Kawazoe Takahashi pairing-friendly hyperelliptic curves with Cocks-Pinch method construction.

Require: $k \in \mathbb{Z}$
Ensure: A genus 2 hyperelliptic curve defined by $y^2 = x^5 + a\ x$ with Jacobian subgroup order ℓ.
Choose a prime number ℓ such that: $lcm(k, 8)$ divides $(\ell - 1)$
Choose α, β and γ such that: α is a primitive k^{th} root of unity in $(\mathbb{Z}/l\ \mathbb{Z})^{\times}$, $\beta^2 \equiv -1 \pmod \ell$ and $\gamma^2 \equiv 2 \pmod \ell$.
Compute c and d integers such that:
$c \equiv (\alpha + \beta)(\gamma(\beta + 1))^{-1} \pmod \ell \quad and \quad c \equiv 1 \pmod 4$,
$d \equiv (\alpha\ \beta + 1)(2\ (\beta + 1))^{-1} \pmod \ell$
Compute $p = c^2 + 2\,d^2$
if (p is a prime satisfying $p \equiv 1 \pmod 8$) **then**
    Compute a such that:
    $a^{(p-1)/2} \equiv -1 \pmod p$,
    $2\ (-1)^{(p-1)/8}\ d \equiv (a^{(p-1)/8} + a^{3(p-1)/8})\ c \pmod p$.
else
    Go to step **3:**
end if
return k, p, ℓ, a, d, c.

4 Group Structure of Hyperelliptic Curve

4.1 General Group Element

The group structure of a hyperelliptic curve $\mathcal{H}$ over a finite field $\mathbb{F}_q$, $(q = p^k)$, p : prime odd number $(p \geq 5)$ and k integer $(k \geq 1)$, recapitulates on the representation of his Jacobian. The following theorem show the unique representation of the Jacobian element.

Definition 2. *A divisor D is a finite formal sum of points on the curve $\mathcal{H}$ such that:*

$$D := \sum_{i=1}^{m} (P_i) \tag{3}$$

We can also define the **reduced divisor** D_r associated with D as follows:

Theorem 1. *Let D an element on the Jacobian of the curve $\mathcal{H}$, the element D has a **unique** representation D_r of the form:*

$$D_r := \sum_{i=1}^{m} (P_i) - m(\infty), \tag{4}$$

such that:

1. *$m \neq g$,*
2. *P_i are affine points,*
3. *The involution $\imath$, satisfy: $\imath(P_i) \neq \imath(P_j)$, for all i, j such that $i \neq j$.*

All elements on Jacobian form an abelian variety, Mumford [13] introduced a way of representing such that elements, it's extremely useful for implementation.

Theorem 2 *(Mumford representation). Let $\mathcal{H}$: $y^2 + h(x)\, y = f(x)$ a hyperelliptic curve of genus g define over a finite field $\mathbb{F}_q$, $q = p^k$ p: primer $(p \geq 5)$ and k integer $(k \geq 1)$. Let K an algebraic extension of $\mathbb{F}_q$, then any element D of the Jacobian of the curve $\mathcal{H}$, can be represented in a **unique** way by two polynomials $(u(x), v(x)) \in K[x]^2$ such that:*

1. *u is monic, with $deg(u(x)) \leq g$,*
2. *$deg(v(x)) \leq deg(u(x))$, and*
3. *$u(x)$ divides $\{v(x)^2 + v(x)\, h(x) - f(x)\}$.*

As we saw before, we have two representations for a divisor D on the Jacobian, a natural representation $D = \sum_{i=1}^{m} (P_i) - m(\infty)$ and a Mumford representation $D = (u(x), v(x))$. To do implementation, we must understand how to manipulate the two representations. For example to pass from the Mumford representation to the natural one, we compute the coordinates of the points $P_i = (x_i, v(x_i))$ with x_i the roots of the polynomial $u(x)$. In the case of genus 2 hyperelliptic curve $\mathcal{H}$, the Mumford representation of divisor of degree 0 whose effective part E is the sum of the points $P1 = (x_1, y_1)$ and $P2 = (x_2, y_2)$, $\quad (E = P_1 \pm P_2)$ is obtained by multiplying and dividing by:

$$u(x) = x^2 - (x_1 + x_2)x + x_1 x_2 \quad and \quad v(x) = \frac{y_1 - y_2}{x_1 - x_2}(x - x_1) + y_1. \tag{5}$$

The set of torsion points are the points whose order is finite, which is the case for all element on the Jacobian $Jac_{\mathcal{H}}$ over $\mathbb{F}_q$. The set of ℓ-torsion points is defined as follows:

1. $[\ell]D = \underbrace{D + D + ... + D}_{\ell \text{ terms}} \quad if \quad \ell > 0,$
2. $[\ell]D = -([-\ell]D) \quad if \quad \ell < 0,$
3. $[0]D = (\infty).$

By definition, we say that D is an ℓ-torsion divisor if $[\ell]D = (\infty)$. The subgroup of ℓ-torsion divisors on the Jacobian of hyperelliptic curve $\mathcal{H}$ over a finite field $\mathbb{F}_q$ is denoted by $Jac_{\mathcal{H}}(\mathbb{F}_q)[\ell]$, with $q = p^k$, p: prime number ($p \geq 5$) and k integer ($k \geq 1$).

4.2 Jacobian Subgroup Operations

To do pairing implementation, we need to perform operations on the Jacobian group $Jac_{\mathcal{H}}(\mathbb{F}_{p^k})[\ell]$, Cantor [1] developed and showed efficient algorithms to manipulate elements of the Jacobian $Jac_{\mathcal{H}}[\ell](\mathbb{F}_{p^k})$, by using the Mumford representation, assuming that $h(x) = 0$ and $p \neq 2$. These algorithms was later generalised by Koblitz [9] to remove these conditions.

We will show here the two algorithms implemented in Sage, the first one is to give a semi-reduced divisor D equivalent to $D_s \simeq D_1 + D_2$ from two semi-reduced divisors D_1 and D_2 (represented by Algorithm 2), and the other algorithm is to reduce the divisor semi-reduced D_s (given by Algorithm 2) to obtain a reduced divisor D_r equivalent (represented by Algorithm 3).

Algorithm 2. Divisor Composition.

1: **Require:** $D_1 = [u_1(x), v_1(x)]$ and $D_2 = [u_2(x), v_2(x)]$
2: **Ensure:** $D_s \simeq D_1 + D_2$, $D_s = [u_s(x), v_s(x)]$.
3: Compute: $d_1 = gcd(u_1(x), u_2(x)) = a_1\ u_1(x) + a_2\ u_2(x)$
4: Compute: $d = gcd(d_1,\ v_1(x) + v_2(x) + h(x)) = b_1\ d_1 + b_2\ (v_1(x) + v_2(x) + h(x))$
5: $c_1 \leftarrow b_1\ a_1$, $c_2 \leftarrow b_1\ a_2$, $c_3 \leftarrow b_2$
6: $u_s(x) \leftarrow (u_1(x)\ u_2(x))\ /\ (d^2)$
7: $v_s(x) \leftarrow (c_1\ u_1(x)\ v_2(x) + c_2\ u_2(x)\ v_1(x) + c_3\ (v_1(x)\ v_2(x) + f(x)))\ /\ d\ mod\ (u_s(x))$
8: **return** $[u_s(x), v_s(x)]$.

Algorithm 3. Divisor Reduction.

1: **Require:** $D = [u(x), v(x)]$, semi-reduced divisor.
2: **Ensure:** $D_r = [u_r(x), v_r(x)]$ reduced with $D_r \simeq D$.
3: Compute: $u_r(x) \leftarrow (f(x) - v(x)\ h(x) - v(x)^2)\ /\ u(x)$
4: Compute: $v_r(x) \leftarrow (-h(x) - v(x))\ mod\ u_r(x)$
5: **if** $(deg(u_r(x)) > g)$ **then**
6: $u(x) \leftarrow u_r(x)$,
7: $v(x) \leftarrow v_r(x)$
8: Go to step **3:**
9: **end if**
10: Make $u_r(x)$ monic.
11: **return** $[u_r(x), v_r(x)]$.

5 Our Work

To compute pairing we need Miller algorithm [12] which makes it possible to calculate the function $f_{\ell,D_1}(D_2)$, this algorithm was applied for the elliptic case and quickly it has been generalised on hyperelliptic curves. We define the group law $\oplus$ on the Jacobian $Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$, let D_1 and $D_2 \in Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$, there is a function $h \in \mathbb{F}_p(\mathcal{H})$ with its divisor:

$$div(h_{D_1,D_2}) = D_1 + D_2 - (D_1 \oplus D_2),$$

The main task involved in computing the evaluation $f_{\ell,D_1}(D)$ in D_1, Miller has shown how to efficiently compute it, this function appearing in

$$Div(f_{\ell,D}) = \ell\ D - D_\ell.$$

For $\ell = n + m$, n and m integers, we find:

$$Div(f_{\ell,D}) = Div(f_{n+m,D}) = f_{n,D}.\ f_{m,D}.\ h_{D_n,D_m},$$

With h a function such that:

$$div(h_{D_n,D_m}) = D_n + D_m - \rho(D_n + D_m),$$

$\rho(D_n + D_m)$: the reduced divisor of $(D_n + D_m)$.

This immediately leads to the following algorithm:

Algorithm 4. Miller's Algorithm for hyperelliptic curves

Require: $\ell \in \mathbb{N}$ *and* $D_1, D_2 \in Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$, reduced-divisors with disjoint support.
Ensure: $f_{\ell,D_1}(D_2)$
Write ℓ in binary form: $\ell = \Sigma_{j=0}^{s}\ \ell_j 2^j$, with $\ell_j \in \{0, 1\}$ and $\ell_s = 1$
$D \leftarrow D_1$
$f \leftarrow 1$
for (j from $s - 1$ to 0) **do**
    Compute $D \leftarrow [2]\, D$ and extract $h_{(D,D)}$
    $f \leftarrow f^2 \cdot h_{(D,D)}(D_2)$
    if ($\ell_j == 1$) **then**
        Compute $D \leftarrow D \oplus D_1$ and extract $h_{(D,D_1)}$
        $f \leftarrow f \cdot h_{(D,D_1)}(D_2)$
    end if
end for
return f

We can clearly see that the execution of the Miller algorithm requires the existence of an algorithm that allows the evaluation of the function $h \in \mathbb{F}_q(\mathcal{H})$, $q = p^k$ p: primer ($p \geq 5$) and k integer ($k \geq 1$). We called this algorithm

"evaluatefunction()", it's the crucial step of Miller's algorithm, it allows to calculate the value of the function in a point $D \in Jac_{\mathcal{H}}(\mathbb{F}_q)$, such that D is a reduced divisor represented in Mumford representation. For this work, we will focus only on the evaluation of the function h in an effective divisor that we note $E = [u_E(x), v_E(x)]$. There are two different methods to compute $h(E)$ (we use in general a norm computation and resultants).

The first method requires a polynomial factorisation of $u_E(x)$, it can be summarized by the following algorithm:

Algorithm 5. Method **1**, function evaluation of h in E.

1: $E \leftarrow \Sigma_{i=1}^{i=d}(P_i)$, $P_i = (x_i, y_i) \in \mathcal{H}$, $D = E - d(\infty)$.
2: Compute the support of E.
3: Factoring $u_E(x)$, as $u_E(x) = \prod_{i=1}^{i=d}(x - x_i)$
4: Setting $y_i = v_E(x_i)$.
5: Note that $(x_i, y_i) \in \mathbb{F}_{q^{g_i}}$, $\quad with \quad g_i \leq g$.
6: Compute $h(E) = \prod_{i=1}^{i=d} h(x_i, y_i) = h(x_1, y_1) \times h(x_2, y_2) \times ... \times h(x_d, y_d)$.

The above method is not the best because it didn't take in consideration the fact that the result of the evaluation has to be in $\mathbb{F}_q$ Instead, one could partition the support into distinct Galois orbits as follows:

$$\{(x_i, y_i), (x_i^q, y_i^q), ..., (x_i^{q^{g_i-1}}, y_i^{q^{g_i-1}})\}$$

And the last step **(6.)** of the algorithm is simply reduced by calculating the norm $N_{\mathbb{F}_{q^{g_i}}/\mathbb{F}_q}(h(x_i, y_i))$.

The second method is faster than the first one, since it does not require any polynomial factorisation. It is based on the observation of $\tilde{h}(x) = h(x, v_E(x))$ which verified for all x_i root of $u_E(x)$, $\tilde{h}(x_i) = h(x_i, v_E(x_i))$ so instead of calculating the product $h(E) = \prod_{i=1}^{i=d} h(x_i, y_i)$, the problem is reduced to the computation of $h(E) = \prod_{i=1}^{i=d} \tilde{h}(x_i)$, with x_i the zeros of $u_E(x)$, but this corresponds exactly to the definition of the resultant of the two polynomials $u_E(x)$ and $\tilde{h}(x)$, and we can write:

$$h(E) = Resultant(u_E(x), h(x, v_E(x)))$$

As we work on hyperelliptic curves of genus g, the polynomials degree $deg(u_E(x))$ of the Mumford representation of $E = [u_E(x), v_E(x)]$ is smaller than g, so we can write:

$$h(E) = Resultant(u_E(x), \tilde{h}(x) \ mod \ u_E(x))$$

We consider $\mathcal{H}$ a hyperelliptic curve of genus 2, defined over a finite field $\mathbb{F}_q$ by $y^2 + h_x y = f_x$, $(q = p^k)$, p : prime odd number $(p \geq 5)$ and k integer $(k \geq 1)$, let $D,\ D_1\ and\ D_2 \in Jac_{\mathcal{H}}(\mathbb{F}_q)$, $D = E - d(\infty)$, E effective divisor.

As $h_{D_1,D_2} = h(x,y)$ is a rational function $h(x,y) \in \mathbb{F}(\mathcal{H})$, we can write:

$$h(x,y) = \frac{h_1(x,y)}{h_2(x,y)}$$

So,

$$\tilde{h}(x) = h(x, v_E(x)) = \frac{h_1(x, v_E(x))}{h_2(x, v_E(x))} = \frac{\tilde{h_1}(x)}{\tilde{h_2}(x)}$$

Algorithm evaluation of the rational function $h = h_{D_1,D_2}(D)$ in E is given by:

Algorithm 6. Method **2**, evaluation of the function h_{D_1,D_2} in E

Require: $E = [u_E(x), v_E(x)]$,$D_1 = [u_1(x), v_1(x)]$ and $D_2 = [u_2(x), v_2(x)]$,
f_x, h_x, $d = deg(u_E(x))$.
Ensure: $h_{D_1,D_2}(E)$.
$\tilde{h_1} \leftarrow u_2(x) \ mod \ u_E(x)$, $\tilde{h_2} \leftarrow 1$, $\tilde{h_3} \leftarrow 1$
$D = [u, v] = D_1 + D_2$, divisors composition D_1 and D_2
while degree of $u > g$ **do**
    $u \leftarrow (f_x - v\ h_x - v^2)/u$
    $v \leftarrow (-h_x - v) \quad mod \quad u$
    Make u monic.
    $\tilde{h_1} \leftarrow (\tilde{h_1}(v_E - v) \ mod \ u_E$
    $\tilde{h_2} \leftarrow (\tilde{h_2}.u) \ mod \ u_E$
    if degree of $v > g$ **then**
        $\tilde{h_3} \leftarrow -\tilde{h_3} \times coef$, $coef$: the leading coefficient of the polynomial $v(x)$.
    end if
end while
Compute R_1 : resultant of the two polynomials $u_2(x)$ and $\tilde{h_1}$
Compute R_2 : resultant of the two polynomials $u_2(x)$ and $\tilde{h_2}$
$\tilde{h_3} = \tilde{h_3}^{\,d}$
return $\frac{R_1}{\tilde{h_3}.R_2}$.

6 Final Exponentiation

Tate pairing algorithm requires computation of final exponentiation after the Miller loop. The optimisation of this computation is to factor the term $(p^k-1)/\ell$ combined with the p-th power Frobenius operations. So the final exponentiation can be written as

$$\frac{p^k - 1}{\ell} := \frac{\phi_k(p)}{\ell} \cdot \prod_{s|k,\ s<k} \phi_s(p) \tag{6}$$

We note that this exponent is determined by fixed system parameters. This final exponent can be broken down into three components. Let $e = \dfrac{k}{2}$ then

$$\frac{p^k - 1}{\ell} := (p^e - 1) \cdot [\frac{(p^e + 1)}{\phi_k(p)}] \cdot [\frac{\phi_k(p)}{\ell}] \tag{7}$$

For example for $k = 28$ the final exponent becomes

$$\frac{p^{28}-1}{\ell} = (p^{14}-1)\cdot[\frac{(p^{14}+1)}{\phi_{28}(p)}]\cdot[\frac{\phi_{28}(p)}{\ell}]$$

With $\phi_{28}(p) = p^{12} - p^{10} + p^8 - p^6 + p^4 - p^2 + 1$, so

$$\frac{p^{28}-1}{\ell} = (p^{14}-1)\cdot(p^2+1)\cdot[\frac{(p^{12}-p^{10}+p^8-p^6+p^4-p^2+1)}{\ell}]$$

There are two parts of the exponentiation, the first one is an easy exponentiation to the power of $exp_1 = (p^{14}-1)\cdot(p^2+1)$ (because of the Frobenius), it also simplifies the rest of the final exponentiation because after raising to the power $(p^{14}-1)$ the field element becomes "unitary". The other part $exp_2 = (p^{12} - p^{10} + p^8 - p^6 + p^4 - p^2 + 1)/\ell$ is the very hard part of the final exponentiation can be calculated using a fast multi-exponentiation algorithm [5]. However, we can use the polynomial description of $p(z)$ and $\ell(z)$ given by Kachisa in [7]. In this case the hard part of the final exponentiation is to the power of $(p^{12} - p^{10} + p^8 - p^6 + p^4 - p^2 + 1)/\ell$. After substituting the polynomials for $p(z)$ and $\ell(z)$, after it can be expressed to the base p.

7 Implementation Results

We have implemented the Tate pairing for the different level security on ordinary genus two curves in SageMath version 8.1. Our aim was not to provide an optimal ad-hoc implementation for any one of the curves or pairings, but rather to keep a sufficient level of security appropriate for a general purpose system, while still implementing algorithmic optimisations that apply in a broader context. All were performed on Intel Core i5-7300HQ CPU @ 2.50 GHz processor.

In the following, we will compute the execution time needed to calculate Tate's pairing for different embedding degree and several levels of security. The following Table 1 shows the calculation time in milliseconds of all Tate's large pairing computation steps on ordinary Kawazoe curves of type $y^2 = x^5 + a\ x$. So we compute the times: t_g, t_J, t_p, t_r, t_m and t_e such that: t_g: time generation of the curve equation, t_J: time computation of the Jacobian on $\mathbb{F}_q$, t_p: Construction time of Jacobian two points, t_r: reducing time of the two divisors, t_m: execution time of the Miller loop, t_e: time required for the final exponentiation.

For t_g, t_J and t_p, we will directly give the time needed by predefined algorithms in Sage to generate the desirable curves, compute Jacobian and to construct two points on the Jacobian $Jac_{\mathcal{H}}(\mathbb{F}_p)$ over prime finite field. On the other hand, the times t_r and t_m are the conclusion of the implementation of the different executable algorithms proposed by Galbraith [2], Granger et al. [4] and others.

To vary the embedding degree, we will always work on genus two Kawazoe and Takahashi curves of type $y^2 = x^5 + a\ x$ generated by Kachisa [7], these parameters are chosen in order to have the desirable ordinary pairing-friendly curves.

Table 1. Execution times in milliseconds (ms) to compute Tate's pairing.

	k = 7	k = 8	k = 10	k = 28
p (bits)	336	387	378	379
ℓ (bits)	254	257	249	255
t_g	1.163	1.228	1.187	1.779
t_J	0.084	0.076	0.033	0.083
t_p	18112,942	14364,803	44049.245	221412.49
t_r (1 D)	0.294	0.337	0.381	1.042
t_m	808.031	10555.84	1221.53	5163.32
t_e	32.1519	57.774	97.863	1561.75

For cryptography applications, the discrete logarithm problems in $Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$ and in the multiplicative group $\mathbb{F}_{p^k}$ must both be computationally infeasible. For Jacobian varieties of hyperelliptic curves of genus 2 the best known discrete logarithm problem (DLP) algorithm is the parallelized rho-Pollard algorithm in [14] and [15], which has running time $O(\sqrt{\ell})$ where ℓ is the size of the largest prime-order subgroup of $Jac_{\mathcal{H}}(\mathbb{F}_{p^k})$. In the following Table 2, we will give the security level for genus two curves according to the size of the curve parameters (k, p, ℓ) and the $\rho - value = \dfrac{g\ log(p)}{log(\ell)}$.

Table 2. Embedding degrees for hyperelliptic curves of genus $g = 2$ required to obtain commonly desired levels of security.

Security level (bits)	Subgroup size (ℓ)	Extension field size (p^k)	Embedding degree(k)			
			$\rho \simeq 1$	$\rho \simeq 2$	$\rho \simeq 3$	$\rho \simeq 4$
80	160	1024	12	6	4	3
128	256	3072	24	12	8	6
192	384	7680	40	20	13	10
256	512	15360	60	30	20	14

Now, we calculate the execution time in Sage needed to compute Tate pairing for different security levels (128, 192 and 256 bits), the following Table 3 lists the execution time of Miller loop and final exponentiation required to compute pairing in milliseconds (ms).

Table 3. Execution times in milliseconds (ms) required to compute Tate's pairing for different security levels.

Security level (bits)	128	192	256
Miller loop	1055.840	1221.53	5163.32
Final exponentiation	57.774	97.863	1561.75
Total	1113.614	1319.393	6725.07

8 Conclusions

In this work, we discuss an implementation of pairings over pairing-friendly hyperelliptic curves. In particular, we focus on Kawazoe-Takahashi genus 2 curves of the form $y^2 = x^5 + a\ x$. We provide the necessary background to have sufficient understanding of pairings on hyperelliptic curves, discuss the algorithm to sample curves of the desired type, and describe the group structure of these curves. We then continue and present details of the Miller algorithm that are involved in the efficient evaluation of the pairing.

First, we present the analogue of the Cocks-Pinch method to obtain ordinary Kawazoe-Takahashi pairing-friendly curves using approach of Kachisa to have curves with a small $\rho - value$, we have implemented this method in Sage, the ordinary Jacobian order over $\mathbb{F}_p$, $\mathbb{F}_{p^k}$ and the various curve parameters are calculated.

Second, we gave several techniques for pairing computation more precisely for Tate pairing case, operations on Jacobian subgroup, Miller loop and we have provided explicit formulae for the evaluation of the function $f_{D_1,D_2}(E)$ in effective divisor required by the Miller algorithm. We then continue and give a performance method for final exponentiation in order to speed up the pairing, generally applicable and which is calculated in two parts, an easy part given a unitary field element and a hard part using polynomial description of curve parameters.

Finally, we gave the implementation results in Sage for different levels of security according to the embedding degree of the curve. Our studies indicates that pairing on hyperelliptic curves is computable and we can have pairing applications efficient and competitive to the pairing on elliptic curves in performance and security level.

As the main contribution here is the evaluation of the rational function h in the point on the Jacobian of the curve over prime field, that appears in the evaluation of $f_{\ell,D_1)}(D)$ in algorithm of Miller. We present one method, that is based on the factorization of polynomials, and a faster one, that instead of factorization of polynomials is based on the resultant of polynomials The includes some interesting ideas to improve the evaluation of pairings on hyperelliptic curves. We also gave a fast method to calculate the final exponentiation by using the parametrization of the curve parameters p and l.

References

1. Cantor, D.G.: Computing in the Jacobian of a hyperelliptic curve. Math. Comput. **48**(177), 95–101 (1987)
2. Galbraith, S.D., Hess, F., Vercauteren, F.: Hyperelliptic pairings. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 108–131. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_7
3. Gaudry, P., Harley, R.: Counting points on hyperelliptic curves over finite fields. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 313–332. Springer, Heidelberg (2000). https://doi.org/10.1007/10722028_18
4. Granger, R., Hess, F., Oyono, R., Thériault, N., Vercauteren, F.: Ate pairing on hyperelliptic curves. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 430–447. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_25
5. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_34
6. Joux, A.: A one round protocol for Tripartite Diffie–Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000). https://doi.org/10.1007/10722028_23
7. Kachisa, E.J.: Generating more kawazoe-takahashi genus 2 pairing-friendly hyperelliptic curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 312–326. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17455-1_20
8. Kawazoe, M., Takahashi, T.: Pairing-friendly hyperelliptic curves with ordinary Jacobians of Type $y^2 = x^5 + ax$. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 164–177. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_12
9. Koblitz, N.: Hyperelliptic cryptosystems. J. Cryptol. **1**(3), 139–150 (1989)
10. Manin, J.I.: The Hasse-Witt matrix of an algebraic curve. In: Selected Papers of Yu I Manin, pp. 3–22. World Scientific (1996)
11. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inf. Theory **39**(5), 1639–1646 (1993)
12. Miller, V., et al.: Short programs for functions on curves. **97**(101–102), 44 (1986, Unpublished manuscript)
13. Mumford, D.: Tata Lectures on Theta i, ii. Birkhäuser, Boston (1984)
14. Pollard, J.M.: Monte carlo methods for index computation. Math. Comput. **32**(143), 918–924 (1978)
15. Van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. J. Cryptol. **12**(1), 1–28 (1999)
16. Yui, N.: On the Jacobian varieties of hyperelliptic curves over fields of characteristic $p > 2$. J. Algebra **52**(2), 378–410 (1978)