Fast computation of Tate pairing on general divisors of genus 3 hyperelliptic curves

* Eunjeong Lee[a], Hyang-Sook Lee[b] and Yoonjin Lee[c]

[a] *School of Computational Sciences, Korea Institute for Advanced Study, 130-722, Seoul, Korea*
[b] *Department of Mathematics, Ewha Womans University, 120-750, Seoul, Korea*
[c] *Department of Mathematics, Simon Fraser University, V5A 1S6, British Columbia, Canada*

Abstract

For the Tate pairing computation over hyperelliptic curves, there are developments by Duursma-Lee and Barreto et al., and those computations are focused on *degenerate* divisors. As divisors are not degenerate form in general, it is necessary to find algorithms on *general* divisors for the Tate pairing computation. In this paper, we present two efficient methods for computing the Tate pairing over divisor class groups of the hyperelliptic curves $y^2 = x^p - x + d,\ d = \pm 1$ of genus 3. First, we provide the *pointwise* method, which is a generalization of the previous developments by Duursma-Lee and Barreto et al. In the second method, we use the *resultant* for the Tate pairing computation. According to our theoretical analysis of the complexity, the *resultant* method is 48.5% faster than the pointwise method in the best case and 15.3% faster in the worst case, and our implementation result shows that the *resultant* method is much faster than the pointwise method. These two methods are completely general in the sense that they work for general divisors with Mumford representation, and they provide very explicit algorithms.

keywords: Tate pairing; hyperelliptic curves; divisors; resultant; pairing-based cryptosystem

Introduction

In recent years the Tate pairing and the Weil pairing have been getting a lot of attentions for designing various protocols in cryptosystem [4, 5, 14, 13, 22, 25, 26]. It is therefore important to develop an efficient implementation of the pairings for the practical applications. The recent papers by Barreto et al. [1] and Galbraith et al. [11] provided the fast computation of the Tate pairing over the supersingular elliptic curves $y^2 = x^3 - x \pm 1$ in characteristic three. In 2003, Duursma and Lee [9] provided a closed formula for the efficient computation of the Tate pairing on $y^2 = x^p - x \pm 1,\ p = 3 \pmod 4$ in characteristic p. After then, Barreto et al. [2] improved Duursma and Lee's result and proposed a general technique for the efficient computation of the Tate pairing on supersingular abelian varieties using *Eta pairing* approach [2, 20]. They also described efficient pairing algorithms on $\mathbb{F}_q$-rational points for elliptic and hyperelliptic curves over $\mathbb{F}_q$.

*(e-mail) ejlee@kias.re.kr, hsl@ewha.ac.kr, yoonjinl@sfu.ca The authors[a,b] were supported by KOSEF, grant number R01-2005-000-10713-0. This work was also done while the last author[c] was visiting KIAS in Seoul during the summers of 2004 and 2005. The author expresses their gratitude to the institute.

In fact, generally divisor operations over hyperelliptic curves are more complicated than point operations over elliptic curves. Therefore, it has been pointed out that *Elliptic Curve Cryptosystem* (ECC) is more efficient than *Hyperelliptic Curve Cryptosystem* (HCC) [25]. The Tate pairing computation uses the Miller algorithm, and the Miller algorithm relies on divisor operations. Thus one expects that the Tate pairing computation over hyperelliptic curves may not be as efficient as that over elliptic curves. However, in some special cases, it was shown that HCC can be made more efficient than ECC by giving the explicit formula for divisor operations [6, 19, 23]. For the higher genus, preserving the same security level, we can decrease the size of the defining field. In fact, some examples given in [6, 19] show that for the efficiency of cryptosystems, the size of the defining field is more important than the complexity of group operation formula. For the Tate pairing, Barreto et al. [2] presented implementation results over elliptic curves and hyperelliptic curves of genus 2, where both are defined over $\mathbb{F}_{2^n}$. The running time for the Tate pairing over hyperelliptic curves was faster than that of elliptic curves. For the Tate pairing computation it is therefore certainly worthwhile to work over some special types of hyperelliptic curves.

Recent developments [2, 9] on the Tate pairing computation on hyperelliptic curves over a finite field $\mathbb{F}_q$ have focused on the case of *degenerate divisors*. However, in the pairing-based cryptography, the efficient Tate pairing implementation over *general divisors* is significantly more important. For instance, in the Boneh-Franklin identity-based encryption scheme, the private keys are general divisors, and therefore the decryption process requires computing a pairing of general divisors. For the case of genus 2, the result in [6] presents both divisor-wise and pointwise approach, and it turns out that the divisor-wise approach is more efficient than the pointwise approach. For the case of genus ≥ 3, when divisors are *general*, there has been no Tate pairing computation method developed so far.

In this paper, we develop general methods of computing the Tate pairing for the genus 3 case. We present two feasible methods by pointwise approach and the resultant approach for computing the Tate pairing over divisor class groups of the hyperelliptic curves $H_d : y^2 = x^p - x + d$, where $d = \pm 1$ and $p = 7$. Those methods are completely general in the sense that they work for general divisors with Mumford representation [21], and we give very explicit and feasible algorithms over H_d. Furthermore, we analyze the complexities of two methods and show implementation results of proposed algorithms. For our two methods, we use the *Eta pairing* for reducing the computation cost. We used SINGULAR [12] software package for symbolic computations in this paper.

In Section 2, we present the pointwise method for computing the Tate pairing over *general* divisors. This method is a generalization of the pointwise method developed in [2] and [9]. In Section 3, we use *resultant* to compute the Tate pairing with Mumford representation. In Section 4, we compare the complexities of two methods and show that the resultant method is 48.5% faster than the pointwise method in the best case and 15.3% faster in the worst case. In Section 5, we report experiment results based on our implementation using NTL [24] software package. According to the implementation, we conclude that the Tate pairing computation by using the resultant is much faster than the pointwise approach. We point out that this is the first implementation over a genus 3 curve.

1 Tate pairing on divisors

Let $\mathbb{F}_q$ be a finite field with q elements, and $H/\mathbb{F}_q$ be a hyperelliptic curve over $\mathbb{F}_q$. We denote by J_H the group of degree zero divisor classes of H. Note that each divisor class can be uniquely

represented by the *reduced divisor* using the *Mumford representation* [21]. Reduced divisors of the curve H can be found as discussed in [17] and [21], and most of reduced divisors in J_H with genus 3 are written as $D = [U_D, V_D] = [x^3 + u_{D,2}x^2 + u_{D,1}x + u_{D,0},\ v_{D,2}x^2 + v_{D,1}x + v_{D,0}]$.

We recall the definition of the Tate pairing [10]. Let ℓ be a positive divisor of the order of $J_H(\mathbb{F}_q)$ with $\gcd(\ell,\ q) = 1$, and k be the smallest integer such that $\ell \mid (q^k - 1)$; such k is called *the embedding degree*. Let $J_H[\ell] = \{D \in J_H \mid \ell D = \mathcal{O}\}$. The *Tate pairing* is a map

$$\begin{aligned} \langle\ \cdot\ \rangle_\ell : J_H[\ell] \times J_H(\mathbb{F}_{q^k})/\ell J_H(\mathbb{F}_{q^k}) &\rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^\ell \\ \langle D, E\rangle_\ell &= f_D(E'), \end{aligned}$$

where $\mathrm{div}(f_D) = \ell D$ and $E' \sim E$ with $support(E') \cap support(\mathrm{div}(f_D)) = \emptyset$. We define the Tate paring value by $t(D,E) = \langle D, E\rangle_\ell^{\frac{q^k-1}{\ell}}$ so that the pairing value is defined uniquely. We write $x^{(i)}$ for x^{p^i}.

We consider a hyperelliptic curve H_d over $\mathbb{F}_q$ defined by $y^2 = x^p - x + d$, $d = \pm 1$, for $p \equiv 3 \pmod 4$, where $q = p^n$ with $\gcd(2p, n) = 1$, and we let $F/\mathbb{F}_q$ and $K/\mathbb{F}_q$ be the extensions of degree $[F : \mathbb{F}_q] = p$ and degree $[K : \mathbb{F}_q] = 2p$, respectively. Over the extension field K, the curve is the quotient of a hermitian curve, hence it is Hasse-Weil maximal. And the class group over K is annihilated by $p^{pn} + 1$; this can be also seen from the following Lemma 1.1. It shows that for $P \in H_d(K)$, $(p^{pn} + 1)((P) - (\mathcal{O}))$ is principal [8].

Lemma 1.1 ([8]). *Let $P = (\alpha, \beta) \in H_d$. The function*

$$h_P = \beta^p y - (\alpha^p - x + d)^{\frac{p+1}{2}}$$

has divisor $(h_P) = p(P) + (P') - (p+1)\mathcal{O}$, where $P' = (\alpha^{(2)} + 2d, \beta^{(2)})$.

From Lemma 1.1, we observe that $p((P)-(\mathcal{O})) \equiv ([p]P)-(\mathcal{O})$, thus the multiplication by p over H_d has an extremely special form such as $[p] = \phi\pi^2$, where $\phi = (x + 2d, -y)$ and π is a Frobenius map of p^{th} power.

Let $m = (n+1)/2$, then from [8] we have

$$|J_{H_{+1}}(\mathbb{F}_q)| = (1+p^n)^3 + (\frac{p}{n})p^m(1+p^n+p^{2n}),\ |J_{H_{-1}}(\mathbb{F}_q)| = (1+p^n)^3 - (\frac{p}{n})p^m(1+p^n+p^{2n}). \quad (1)$$

1.1 Eta pairing on H_d

We discuss the *Eta pairing* introduced in [2] which is very useful for efficient computation of the Tate pairing. We consider a hyperelliptic curve H_d over some finite field $\mathbb{F}_{p^n}$, and let ψ be an endomorphism on the curve H_d given by

$$\psi : H_d(K) \rightarrow H_d(K), \quad \psi(x, y) = (\rho - x, \sigma y), \quad (2)$$

where $\rho \in F$ is a root of $\rho^p - \rho + 2d = 0$, and $\sigma, \bar{\sigma} \in K$ are the roots of $\sigma^2 + 1 = 0$.

For efficient Tate pairing computation, we concern with *the twisted Tate pairing*

$$\begin{aligned} \hat{t}\ :\ J_{H_d}[\ell] \times J_{H_d}(\mathbb{F}_{p^{2pn}})/\ell J_{H_d}(\mathbb{F}_{p^{2pn}}) &\longrightarrow \mathbb{F}^*_{p^{2pn}} \\ \hat{t}(D,\ E) &= f_D(\psi(E))^{p^{pn}-1}, \end{aligned}$$

where $(f_D) = (p^{pn}+1)D$ from [9, Theorem 4].

For two divisors D and E in J_{H_d}, the *Eta pairing* is defined by

$$\eta(D,E) = \prod_{i=0}^{n-1} h_{D_i}(\psi(E))^{p^{n-1-i}}, \tag{3}$$

where $D_{i+1} + (h_{D_i}) = pD_i$ with a divisor $D_0 = D$ and some rational function h_{D_i}. By Lemma 1.1, H_d has a property that

$$q(P) - q(\mathcal{O}) = (\gamma(P)) - (\mathcal{O}) + (g_P), \tag{4}$$

for some automorphism γ on H_d and some function g_P. Thus γ can be given by $\gamma = \phi^n\pi^{2n}$. The following theorem is crucial for efficient computation of the Tate pairing on divisors.

Theorem 1.2 ([2], [20]). *Let $q = p^n, p \equiv 3 \pmod 4$, $\gamma = \phi^n\pi^{2n}$ on J_{H_d} induced from Eq. (4), and ψ be an endomorphism on the curve H_d over $\mathbb{F}_q$. Assume that*

$$\phi^n\psi^{[q]} = \psi, \tag{5}$$

where $\psi^{[q]}$ denotes a map obtained by raising the coefficients of ψ by q^{th} power. Then for divisors D and E in $J_H(\mathbb{F}_q)$, we have $\eta(qD,E) = \eta(D,E)^q$.

For any divisor $E = [U_E, V_E]$ in $J_{H_d}(\mathbb{F}_q)$, the endomorphism ψ in Eq. (2) on divisors are easily deduced as follows: $\psi(E) = [U_{\psi(E)},\ V_{\psi(E)}]$, where

$$\begin{aligned} U_{\psi(E)} &= x^3 - (3\rho + u_{E,2})x^2 + (3\rho^2 + 2u_{E,2}\rho + u_{E,1})x - (\rho^3 + u_{E,2}\rho^2 + u_{E,1}\rho + u_{E,0}),\\ V_{\psi(E)} &= \sigma(v_{E,2}x^2 - (2\rho v_{E,2} + v_{E,1})x + v_{E,2}\rho^2 + v_{E,1}\rho + v_{E,0}). \end{aligned} \tag{6}$$

Using Eq. (6), a straightforward verification shows that H_d satisfies the crucial condition in Eq. (5). From Theorem 1.2, it thus follows that

$$\hat{t}(D,\ E) = \eta(D,E)^{7^{6n+1}(7^{7n}-1)}. \tag{7}$$

It is therefore enough to compute $\eta(D,E)$ to obtain $\hat{t}(D,\ E)$ for any divisors $D, E \in J_{H_d}(\mathbb{F}_q)$. When all the points in *support*(D) and *support*(E) are $\mathbb{F}_q$-rational points, using Eq. (7) makes the Tate pairing computation very efficient as mentioned in [2]. In Section 2, we will extend the concept of the Eta pairing on the general divisors, that is, the supports of D and E are not necessarily $\mathbb{F}_q$-rational points.

Throughout this paper, we focus on the hyperelliptic curve $H_d : y^2 = x^p - x + d,\ d = \pm 1, p \equiv 3 \pmod 4$ of genus $g = 3$, therefore we work on the case $p = 7$; this case is cryptographically useful [9].

2 Pointwise computation of the Tate pairing

This section presents a generalization of the pointwise method developed in [2] and [9], and this method can be used for any divisors, not only for $\mathbb{F}_q$-rational points. The Eta pairing is used for reducing the computation cost as well.

As mentioned in Eq. (7), for divisors D, E in J_{H_d}, the Tate pairing can be computed by

$$\hat{t}(D,\ E) = \eta(D,E)^{7^{6n+1}(7^{7n}-1)}, \tag{8}$$

where D and E have the form $D = (P_1) + (P_2) + (P_3) - 3(\mathcal{O}),\ \ E = (Q_1) + (Q_2) + (Q_3) - 3(\mathcal{O})$ for points P_k and Q_j contained in $H_d(\mathbb{F}_{7^{3n}})$ with $k, j = 1, 2, 3$.

From Eq. (3) we have

$$\eta(D, E) = \prod_{i=0}^{n-1} h_{D_i}(\psi(E))^{7^{n-1-i}}.$$

For $i \geq 1$, let $D_i = (P_{i,1}) + (P_{i,2}) + (P_{i,3}) - 3(\mathcal{O})$, then $h_{D_i}(\psi(E))$ can be computed by

$$h_{D_i}(\psi(E)) = \prod_{k,j=1}^{3} h_{P_{i,k}}(\psi(Q_j)), \tag{9}$$

where $(h_{P_{i,k}}) = 7(P_{i,k}) + (P_{i,k}{}') - 8(\mathcal{O}),\ P_{0,k} = P_k,\ P_{i,k} = (\alpha^{(2i)} + i2d, (-1)^i \beta^{(2i)}) \sim 7^i[P_{0,k}]$ and $[P'_{i,k}] = -[7P_{i,k}]$ for $i \geq 1$.

Algorithm 1 shows the pointwise computation of Tate pairing.

Algorithm 1 Pointwise computation

INPUT: $D, E \in J_{H_d}(\mathbb{F}_{7^n})$
OUTPUT: $\hat{t}(D,\ E)$
1. $g \leftarrow 1$
2. Compute $P_k = (\alpha_k, \beta_k),\ k = 1, 2, 3$ and $Q_j = (x_j, y_j),\ j = 1, 2, 3$
which are supporting points for D and E, respectively.
3. For $i = 0$ to $n - 1$ do
4. For $k, j = 1$ to 3 do
5. compute $h_{k,j} = \beta_k^7 \cdot y_j \cdot \sigma - (\alpha_k^7 + x_j + d - \rho)^4$
6. set $\alpha_k \leftarrow \alpha_k^{7^2} + 2d,\ \beta_k \leftarrow \beta_k^{7^2}$
7. set $g \leftarrow g^7 \cdot \prod_{k,j=1}^{3} h_{k,j}$
8. Return $g^{7^{6n+1}(7^{7n}-1)}$, where $g = \eta(D, E)$.

Let m_3 (resp. M_3, M) be the time cost for a multiplication in $\mathbb{F}_{7^{3n}}$ (resp. $\mathbb{F}_{7^{3(14n)}}$, $\mathbb{F}_{7^{14n}}$). For simplicity, we assume that a squaring cost is similar to a multiplication cost, and we omit the computation cost for 7^{th} powering since it is negligible compared with the other operations.

Step 5 requires two multiplications and two squarings in $\mathbb{F}_{7^{3n}}$, and Step 7 needs eight multiplications in $\mathbb{F}_{7^{3(14n)}}$ and one multiplication in $\mathbb{F}_{7^{14n}}$. For computing $\hat{t}(D,\ E)$, the total complexity is therefore

$$T_P := 2\ T_{3rt} + n(9 \cdot 4m_3 + 8M_3 + 1M) = 2\ T_{3rt} + n\ (36m_3 + 8M_3 + 1M), \tag{10}$$

where T_{3rt} is the time for finding all the roots of a cubic polynomial over $\mathbb{F}_{7^{3n}}$; this is required for obtaining the supporting points of D and E.

3 Computation of the Tate pairing by using the resultant

In this section, our goal is, for given divisor inputs with the divisor representation, to find an efficient algorithm which provides us the final Tate pairing value over H_d. For evaluating a function at a

divisor, we apply the *resultant* for the efficient computation of the Tate pairing, and show that this approach is much faster than the first method.

According to Eq. (7), for the Tate pairing computation over divisors $D, E \in J_H(\mathbb{F}_{7^n})$, it is sufficient to compute $\eta(D, E)$. In the following subsections 3.1 and 3.2, to obtain the value of $\eta(D, E)$, we find the explicit formulas for $D_i = [7^i]D$ and h_{D_i} for $i \geq 1$, and we also obtain the evaluation formula of rational function h_{D_i} at a divisor in a very explicit way.

3.1 7-multiplication on divisors

Let D be a reduced divisor of H_d such that

$$D = (P_1) + (P_2) + (P_3) - 3\mathcal{O} = [U_D, V_D],$$

where $P_j = (\alpha_j, \beta_j)$ for $j = 1, 2, 3$, $U_D = x^3 + u_{D,2}x^2 + u_{D,1}x + u_{D,0}$, and $V_D = v_{D,2}x^2 + v_{D,1}x + v_{D,0} \in \mathbb{F}_{7^n}[x]$. Let $D_0 = D$, $D_{i+1} + (h_{D_i}) = 7D_i$, and $D_i = [U_{D_i}, V_{D_i}]$ for each positive integer i.

The following lemma provides us with explicit formulas for U_{D_i} and V_{D_i} in terms of the coefficients of U_D and V_D for $i \geq 1$. The proof can be obtained from the knowledge of Section 5 in Appendix of [17].

Lemma 3.1. *Let $[7]$ be the multiplication map by 7 on the divisor class group of $H_d/\mathbb{F}_{7^n}$. Then we have, for $i \geq 1$, $[7^i]D = D_i = [U_{D_i}, V_{D_i}]$ with*

$$\begin{aligned} U_{D_i} &= x^3 + (u_{D,2}^{(2i)} + id)x^2 + (u_{D,1}^{(2i)} + 3idu_{D,2}^{(2i)} - 2i^2)x + u_{D,0}^{(2i)} - 2idu_{D,1}^{(2i)} - 3i^2u_{D,2}^{(2i)} - i^3d, \\ V_{D_i} &= (-1)^i v_{D,2}^{(2i)}x^2 + (-1)^i(3idv_{D,2}^{(2i)} + v_{D,1}^{(2i)})x + (-1)^i(-3i^2v_{D,2}^{(2i)} - 2idv_{D,1}^{(2i)} + v_{D,0}^{(2i)}). \end{aligned}$$

In the following proposition, we find the function h_D such that $(h_D) = 7D + D'$ in an explicit way.

Proposition 3.2. *Let $h_D(x, y)$ be a function such that $(h_D) = 7D + D'$, and τ be a map*

$$\tau : H_d \to \hat{H}_d, \;\; (x, y) \to (\hat{X}, Y) = (x - \xi^7 - d, y).$$

Then, for appropriate ξ, we have

$$\hat{h}_{\tilde{D}}(\hat{X}, Y) := h_D(x, y) \circ \tau^{-1} = \delta_1 Y^3 + s(\hat{X})Y^2 + t(\hat{X})Y + \delta_{16}(\hat{X}),$$

where $\delta_{16}(\hat{X}) = -(\hat{X}^3 + \tilde{u}_1^7\hat{X} + \tilde{u}_0^7)^4$, and δ_1, $s(\hat{X})$ and $t(\hat{X})$ are described in Table 5 of Appendix.

Proof. Let $D = (P_1) + (P_2) + (P_3) - 3(\mathcal{O})$, and $7((P_j) - (\mathcal{O})) = (h_j) - (P_j') + (\mathcal{O})$ for $j = 1, 2, 3$. Then

$$7D = (h_1h_2h_3) - [(P_1') + (P_2') + (P_3') - 3(\mathcal{O})] = (h_D) - D',$$

where $D' = (P_1') + (P_2') + (P_3') - 3(\mathcal{O})$. For simplicity, the change of variable

$$\tau_\xi : x \to X = x - \xi, \;\; y \to Y = y$$

with $\xi = -\dfrac{u_{D,2}}{3}$ transforms the curve H_d to a curve $\tilde{H}_d : Y^2 = X^7 - X + (\xi^7 - \xi + d)$.

Let $P_j = (\alpha_j, \beta_j)$, $\tilde{P}_j = (\alpha_j - \xi, \beta_j)$, $\tilde{\alpha_j} = \alpha_j - \xi$ and $\tilde{\beta_j} = \beta_j$. From the fact that $\beta_j = V_D(\alpha_j)$ and $\tilde{\beta_j} = V_{\tilde{D}}(\tilde{\alpha_j}) = V_{\tilde{D}}(\alpha_j - \xi)$, it follows that $U_{\tilde{D}} = x^3 + \tilde{u}_1 x + \tilde{u}_0$ and $V_{\tilde{D}} = \tilde{v}_2 X^2 + \tilde{v}_1 X + \tilde{v}_0$. We also have $\tilde{D} = (\tilde{P}_1) + (\tilde{P}_2) + (\tilde{P}_3) - 3(\mathcal{O}) = [X^3 + \tilde{u}_1 X + \tilde{u}_0,\ \tilde{v}_2 X^2 + \tilde{v}_1 X + \tilde{v}_0]$. Furthermore, $(\tilde{h_j}) = 7(\tilde{P_j}) + (\tilde{P_j}') - 8(\mathcal{O})$, where $\tilde{h_j} = h_j(X + \xi, Y)$. Letting $\theta = \xi^7 - \xi + d$, it is easy to see that $\tilde{h_j} = \tilde{\beta_j}^7 Y - (\tilde{\alpha_j}^7 - X + \theta)^4 = (\tilde{v}_2 \tilde{\alpha_j}^2 + \tilde{v}_1 \tilde{\alpha_j} + \tilde{v}_0)^7 Y - (\tilde{\alpha_j}^7 - X + \theta)^4$. Thus we obtain

$$\begin{aligned} h_{\tilde{D}}(X, Y) &= \tilde{h_1}(X, Y)\tilde{h_2}(X, Y)\tilde{h_3}(X, Y) \\ &= \prod_{j=1}^{3} \left((\tilde{v}_2 \tilde{\alpha_j}^2 + \tilde{v}_1 \tilde{\alpha_j} + \tilde{v}_0)^7 Y - (\tilde{\alpha_j}^7 - X + \theta)^4 \right). \end{aligned} \tag{11}$$

If we apply the Elimination method in [7] to Eq. (11) with elimination order $\{\tilde{\alpha_1}, \tilde{\alpha_2}, \tilde{\alpha_3}\} > \{\tilde{u}_1, \tilde{u}_0\}$, then we can obtain Eq. (11) as a function of $\tilde{u}_1$ and $\tilde{u}_0$. The coefficients for $\hat{h}_{\tilde{D}}(\hat{X}, Y) = h_{\tilde{D}}(X - \theta, Y)$ are described in Table 5, where the second column shows the corresponding coefficient in terms of $\tilde{v_i}'$s and $\tilde{u_i}'$s. □

3.2 Evaluation at a divisor

In this subsection, we use resultant to evaluate a rational function at a divisor, which is necessary to achieve our goal. For the definition of resultant and its properties, we refer to [27, Ch. VI].

Theorem 3.3. *Let F be a field. For $A, B \in F[x]$ with deg $A = m$, deg $B = n$, we have*

$$res(A, B) = a^n \prod_{i=1}^{m} B(\alpha_i),$$

where $\alpha_1, \alpha_2, \cdots, \alpha_m \in \bar{F}$ (= algebraic closure of F) are all the roots of A and a is the leading coefficient of A.

With the same notations as in Theorem 3.3, furthermore, we have

$$res(A, B) = (-1)^{mn} res(B, A). \tag{12}$$

In addition, efficient reduction method for computing the resultant is also introduced in [27, Ch. VI]. When $m \geq n$, by Euclidean division algorithm, there exists $Q(x), R(x) \in F(x)$ such that $A(x) = Q(x)B(x) + R(x)$ with deg $R < n$. Then

$$res(A, B) = (-1)^{mn} res(B, R). \tag{13}$$

Now we are ready to apply the resultant to our Tate pairing computation.

Lemma 3.4. *For $h_D(x, y)$ and $E = [U_E, V_E]$, we let $H_{D,E}(x) = h_D(x, V_E(x))$. Then we have*

$$h_D(E) = res(U_E, H_{D,E}).$$

Proof. In fact, $h_D(E) = H_{D,E}(x_1)H_{D,E}(x_2)H_{D,E}(x_3)$, where x_i's are the roots of $U_E(x)$. The assertion thus follows immediately from Theorem 3.3. □

Algorithm 2 Tate pairing computation by using resultant

INPUT: $D = [U_D,\ V_D],\ E = [U_E,\ V_E] \in J_{H_d}(\mathbb{F}_{7^n})$, endomorphism ψ
OUTPUT: $\hat{t}(D,\ E)$
1. Set $\xi \leftarrow 2u_{D,2}$ and $\theta \leftarrow \xi^7 - \xi + d$.
2. Compute $u_1 = 3\xi^2 + 2\xi u_{D,2} + u_{D,1}$ and $u_0 = \xi^3 + u_{D,2}\xi^2 + u_{D,1}\xi + u_{D,0}$.
3. Compute δ_j for $j = 1, ..., 15$ using Table 5
4. $g \leftarrow 1$,
5. for $i = 0$ to $n - 1$ do
6. compute $\hat{E} = \tau_{\xi+\theta}(\psi(E))$
7. compute $H_{\tilde{D},\hat{E}} = \hat{h}_{\tilde{D}}(x, V_{\hat{E}})$ and $R = H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}}$ (Table 6).
8. compute $h_{D_i}(\psi(E)) = res(U_{\hat{E}}, R)$.
9. $g \leftarrow g^7 \cdot h_{D_i}(\psi(E))$
10. set $u_0 \leftarrow u_0^{7^2},\ u_1 \leftarrow u_1^{7^2}$
11. set $\xi \leftarrow \xi^{7^2},\ \theta \leftarrow \theta^{7^2},\ \delta_j \leftarrow \delta_j^{7^2}$ if $j = 2, 3, 4, 5, 6$, and $\delta_j \leftarrow (-1)^i \delta_j^{7^2}$ otherwise.
12. Return $g^{7^{6n+1}(7^{7n}-1)},\ g = \eta(D, E)$.

Table 6 of Appendix describes the nonzero coefficients of $H_{D,E}(x)$, and the complexity is counted by using *Karastuba's technique* [15].

Now by using the reduction method in Eq. (13), we can compute $res(U_E, H_{D,E})$ as follows:

Lemma 3.5. *Let $H_{D,E}(x) = Q(x)U_E(x) + R(x)$ with deg $R \leq 2$. Then we have*

$$h_D(E) = res(U_E, R).$$

Proof. We observe that the degree of $H_{D,E}$ is 12 and the degree of U_E is 3. Thus by using Eq. (12), we have $res(U_E, H_{D,E}) = res(H_{D,E}, U_E)$. From Eq. (13) it follows that $res(H_{D,E}, U_E) = res(U_E, R)$, so we get $res(U_E, H_{D,E}) = res(U_E, R)$. □

3.3 Algorithm for the Tate pairing computation by using the resultant

In this subsection, we describe an algorithm for computing the Tate pairing on divisors, and we also compute its complexity. From Lemma 3.1, Proposition 3.2 and Lemma 3.4, the Tate pairing given in Eq. (7) can be computed by using Algorithm 2. Since $v_{\hat{E},j} = \sigma \cdot (\text{some element in } \mathbb{F}_{7^{7n}}),\ j = 0, 1, 2$, we note that $H_{\tilde{D},\hat{E}}$ in the step 7 of Algorithm 2 can be written as

$$H_{\tilde{D},\hat{E}} = -x^{12} + \sum_{i=0}^{10} (d_i\sigma + e_i)x^i,\ \ d_i, e_i \in \mathbb{F}_{7^{7n}} \text{ for } 0 \leq i \leq 10.$$

To find $H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}}$ in the step 7, we use the following recursive relations:

$$x^i \equiv a_i x^2 + b_i x + c_i \pmod{U_E},\ \ 3 \leq i \leq 12$$
$$a_3 = -u_{\hat{E},2},\ b_3 = -u_{\hat{E},1},\ c_3 = -u_{\hat{E},0} \in \mathbb{F}_{7^{7n}}$$
$$a_i = a_{i-1}a_3 + b_{i-1},\ b_i = a_{i-1}b_3 + c_{i-1},\ c_i = a_{i-1}c_3.$$

Then R can be computed by

$$
\begin{aligned}
R &= H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}} \\
&= (a_{12} + \sum_{i=3}^{10} a_i(d_i\sigma + e_i) + d_2\sigma + e_2)x^2 + (b_{12} + \sum_{i=3}^{10} b_i(d_i\sigma + e_i) + d_1\sigma + e_1)x \\
&+ (c_{12} + \sum_{i=3}^{10} c_i(d_i\sigma + e_i) + d_0\sigma + e_0).
\end{aligned} \tag{14}
$$

Now we discuss the complexity of Algorithm 2 by counting the number of operations which are necessary for computing $\eta(D, E)$. We denote the time for multiplications in $\mathbb{F}_{7^{14n}}, \mathbb{F}_{7^{7n}}$ and $\mathbb{F}_{7^n}$ by M, m' and m, respectively, and a multiplication between $\mathbb{F}_{7^n}$ and $\mathbb{F}_{7^{7n}}$ by $\tilde{m}$. Noting that, in Step 6, $u_{\hat{E},j}, j = 0, 1, 2$ and $v_{\hat{E},0}, v_{\hat{E},1}$ belong to $\mathbb{F}_{7^{7n}}$, and from Eq. (6) we have $v_{\hat{E},2} = \sigma \cdot$ (some element in $\mathbb{F}_{7^n}$). The computation cost of $H_{\tilde{D},\hat{E}} = \hat{h}_{\tilde{D}}(x, V_{\hat{E}})$ in Step 7 is counted in Table 6. We need $7m + 18m'$ for computing $x^i \pmod{U_E}$ since we do not need the computation x^{11}, and we need $48m'$ to compute R in Eq. (14).

The total complexity of this algorithm is therefore

$$
40m + n(29m + 31\tilde{m} + 70m' + T_{res} + 1M), \tag{15}
$$

where T_{res} is the computation cost for the resultant $res(U_{\hat{E}}, R)$ of $U_{\hat{E}}$ and R in $\mathbb{F}_{7^{14n}}$. In detail, we need $3m$ in the step 2, and $37m$ in the step 3 from Table 5. For each loop, we need $5m$ in the step 6, $17m + 31\tilde{m} + 4m'$ in the step 7 from Table 6, and $1M$ in the step 9.

We calculate the resultant by computing the determinant of two polynomials with degree 2 and 3 in Mumford representation. Then we have $T_{res} = 48m'$, where m' is a multiplication (resp. squaring) in $\mathbb{F}_{7^{7n}}$.

4 Complexity comparison

In this section we compare the complexities of our two methods given in Section 2 and 3.

When an extension degree is of the form $k = 2^i 3^j$, the computation cost for a multiplication in $\mathbb{F}_{q^k}$ is theoretically $3^i 5^j$ times of the cost for a multiplication $\mathbb{F}_q$ ([16], [18]). From this observation, we assume that

$$
1 \text{ mult. in } \mathbb{F}_{7^{3n}}(m_3) \approx 5m, \;\; 1 \text{ mult. in } \mathbb{F}_{7^{3(14n)}}(M_3) \approx 5M, \;\; 1 \text{ mult. in } \mathbb{F}_{7^{14n}}(M) \approx 3m' \tag{16}
$$

and we also let $\tilde{m} \approx 7m$. With the above assumptions, the pointwise computation cost in Eq. (10) is $T_P := 2\ T_{3rt} + n\ (36 \cdot 5m + 123m')$, where T_{3rt} is the time for finding all the roots of a cubic polynomial over $\mathbb{F}_{7^{3n}}$. By *Berlekamp-Rabin algorithm* [3], we have $T_{3rt} = O(3^2 \log 3 \log 7^{3n}) \cdot 5m \approx 27n \cdot 4 \cdot m_3 = 108nm_3$.

Counting the cost for T_{3rt}, we finally have

$$
T_P \approx n(1260m + 123m'). \tag{17}
$$

On the other hand, the total time for the resultant method in Eq. (15) is $T_R := 40m + n(246m + 70m' + T_{res} + 1M)$, where T_{res} is the time for computing the resultant of two polynomials over $\mathbb{F}_{7^{14n}}$. As mentioned in Section 3.3, we have $T_{res} = 48m'$, where m' is a multiplication in $\mathbb{F}_{7^{7n}}$. Thus, the computation cost of our resultant approach is approximately

$$
T_R = 40m + n(246m + 121m'). \tag{18}
$$

Table 1: Complexity comparison: examples

security (bits)	80	128	192
bitlength of 7^{14n}	1140	3072	8192
n	29	79	211
pointwise (T_P)	$36540m + 3567m'$	$99540m + 9717m'$	$265860m + 25953m'$
resultant (T_R)	$7174m + 3509m'$	$19474m + 9559m'$	$51946m + 25531m'$
$\frac{T_P}{T_R}$	$1.17982 \leq \frac{T_P}{T_R} \leq 1.93808$	$1.17998 \leq \frac{T_P}{T_R} \leq 1.93963$	$1.18004 \leq \frac{T_P}{T_R} \leq 1.94019$

To compare the complexities of two methods, we summarize T_P, T_R and the ratio T_P/T_R in Table 1, where the examples are chosen for cryptographically meaningful values [18].

According to [16], the ratio of m' and m is $7 \leq \frac{m'}{m} \leq 49$, and the last row of Table 1 shows the range of the ratio of $\frac{T_P}{T_R}$ for each fixed value of n. For each n, as m'/m is decreasing, the ratio $\frac{T_P}{T_R}$ is increasing. This implies that as the performance of m', the multiplication in $\mathbb{F}_{7^{7n}}$, is getting more efficient, the resultant method gets more efficient than the pointwise method. Furthermore, we observe that when m'/m is fixed, as n is increasing, $\frac{T_P}{T_R}$ is also increasing. Thus, for higher security level, the resultant method gives better efficiency than the pointwise method. Therefore, we can conclude that the Tate pairing computation by using the resultant is 48.5% faster than the pointwise computation in the best case and 15.3% faster in the worst case.

5 Experimental results

We have proposed two methods by the resultant and pointwise approach for computing the Tate pairing over the genus 3 hyperelliptic curve $H_d : y^2 = x^7 - x + d, d = \pm 1$ over $\mathbb{F}_q$. In Section 4, we compared the complexities of two proposed methods, and it turned out that the resultant approach is up to 48.5 % faster than the pointwise approach. In this section, we provide experiment results for the Tate pairing computation over the genus 3 hyperelliptic curve and compare the running time. Basically our goal in this experimentation is that we verify our theoretical complexity comparison in Section 4 by actual implementation using one of the standard packages such as NTL. We make implementations when both input divisors D and E are general divisors, and we use the NTL software package. We provide implementation for genus 3 hyperelliptic curves for the first time.

We first need to find a prime n for each security level s such that $2^s \approx 7^{3n}$, and also find a large prime ℓ dividing $|J_{H_d}(\mathbb{F}_{7^n})|$ such that $\ell \approx 2^s$. The formula for $|J_{H_d}(\mathbb{F}_{7^n})|$ is given in Eq. (1). By searching for good candidates for ℓ and n from $n = 29$ through $n = 79$, we find the four values of n, namely, $29, 43, 47$ and 73 with corresponding primes as given in Appendix.

Table 2 shows the amount of time to perform the field multiplications in $\mathbb{F}_{7^n}$, $\mathbb{F}_{7^{3n}}$ and $\mathbb{F}_{7^{7n}}$ using NTL. The table was computed by taking average time of 5000 multiplications of random elements in each field.

In Section 4, we assumed the ratio m_3/m is 5 for the field operations m_3 in $\mathbb{F}_{7^{3n}}$ and m in $\mathbb{F}_{7^n}$. However, in NTL the actual ratio m_3/m is approximately 7 or 9 as shown in Table 2, and m'/m in Table 2 is in the range $7 \leq m'/m \leq 49$ as we expected. In our implementation M_3 is optimized to $56m_3$. Therefore, each complexity for the Tate pairing computation is given by $T_R = 40m + n(246m + 121m')$ and

$$T_P = 2\ T_{3rt} + n\ (36m_3 + 8M_3 + 1M) = 2T_{3rt} + n(36m_3 + 56 \cdot 8m_3 + 3m') \approx n(700m_3 + 3m').$$

Table 2: Multiplication timings (in milliseconds)

n	29	43	47	73
$\mathbb{F}_{7^n}$ (m)	0.2562	0.5686	0.6626	1.6062
$\mathbb{F}_{7^{3n}}(m_3)$	2.4	4.3218	4.8906	11.1562
$\mathbb{F}_{7^{7n}}$ (m')	7.8438	16.672	17.5156	27.8688
m_3/m	9.36768	7.60077	7.38092	6.94571
m'/m	30.6159	29.3211	26.4347	17.3508

According to the actual ratio of the field operations in NTL, Table 1 is adjusted to obtain Table 3.

As shown in Table 3, for $n \geq 43$, as n increases, T_P/T_R also increases as we expected in Section 4. On the other hand, when n increases from 29 to 43, T_P/T_R decreases, which is opposite to what we expected, and we guess that the reason is the following: For instance, we observe that when n changes from 29 to 43, the decrement of m_3/m (resp. m'/m) is 1.767 (resp. 1.295). On the other hand, when n changes from 43 to 47, the decrement of m_3/m (resp. m'/m) is 0.220 (resp. 2.886). So, the decrement of m_3/m is much larger than m'/m when n changes from 29 to 43, while m_3/m is much smaller than m'/m when n changes from 29 to 43.

From Table 3, the resultant method is 40.57%(resp. 29.38%, 34.32%, and 52.26%) faster than the point-wise method for $n = 29$ (resp. 43, 47, and 73). These examples support our theoretical complexity analysis in Section 4, that is, for higher security level the resultant method is more efficient than the pointwise method for the Tate pairing computation.

Table 3: Complexity comparison: examples in NTL

bitlength of 7^{14n}	1140	1690	1847	2869
n	29	43	47	73
pointwise (T_P)	$20300m_3 + 87m'$	$30100m_3 + 129m'$	$32900m_3 + 141m'$	$51100m_3 + 219m'$
resultant (T_R)	$7174m + 3509m'$	$10618m + 5203m'$	$11602m + 5687m'$	$17998m + 8833m'$
$\frac{T_P}{T_R}$	1.68254	1.42525	1.52257	2.09465

Table 4 shows the implementation results of the Tate pairing for selected examples. The resultant method is 48.8% (resp. 39.1%, 38.7%, and 43.4%) faster than the pointwise method for $n = 29$ (resp. $43, 47$, and 73). We performed fifty calculations with random samples for each method and took the average time. The experiments ran on a machine with 2.2Ghz Opteron, and we used Microsoft Visual C++ 6.0.

Our implementation shows that the performance ratio $\frac{T_P}{T_R}$ for $n = 29$ is larger than the ratio for $n = 73$, while the theoretical complexity analysis in Table 3 shows the other way around. This difference occurs because of the relative time cost $\frac{T_{3rt}}{T_P}$ for computing all the supporting points of an input divisor. In more detail, in the theoretical complexity analysis, for $\frac{T_{3rt}}{T_P}$ we have a flat ratio approximately 0.152. On the other hand, in our implementation, $\frac{T_{3rt}}{T_P}$ is 0.447242 (resp. $0.436616, 0.41533$, and 0.170231) for $n = 29$ (resp. $43, 47$, and 73). Thus the ratio $\frac{T_{3rt}}{T_P}$ is various depending on the values of n, and in fact, the ratio is largest when $n = 29$ and smallest when

Table 4: Experiment results (in seconds)

bit-length of ℓ	237	338	373	608
bit-length of 7^{14n}	1140	1690	1847	2869
n	29	43	47	73
pointwise method(T_P)	38.3564	99.2936	120.672	399.962
resultant method(T_R)	19.6315	60.4622	73.9278	226.412
$\frac{T_P}{T_R}$	1.95382	1.64224	1.63229	1.76652

$n = 73$.

According to the theoretical complexity analysis and the implementation, we conclude that the resultant method is faster than the pointwise method for the Tate pairing computation.

Conclusions. In this paper, we have provided two methods by pointwise approach and resultant approach for computing the Tate pairing over divisor class groups of the hyperelliptic curves $H_d : y^2 = x^p - x + d,\ d = \pm 1$ of $p = 7$. From the theoretical analysis of the complexity, the resultant method is 48.5% faster than the pointwise method in the best case and 15.3% faster in the worst case, and our implementation results also support our theoretical analysis of the complexity. There is room for further optimization in the implementation, and using NTL might not be the best choice. However, our implementation is sufficient enough to conclude that the resultant method is more efficient than the pointwise method in the Tate pairing computation.

References

[1] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, M. Scott, Efficient Algorithms for Pairing-Based Cryptosystems. Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science, Vol. 2442, Springer-Verlag, (2002) 354–368.

[2] P. S. L. M. Barreto, S. Galbraith, C. hEigeartaigh and M. Scott, Efficient Pairing Computation on Supersingular Abelian Varieties, Cryptology eprint Atchives, Available at http://eprint.iacr.org, 2004, No. 2004/375.

[3] E. R Berlekamp, Factoring polynomials over large finite fields, Math. Comp., Vol 24, (1970) 713-735.

[4] D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing. Advances in Cryptology, Crypto 2001, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, (2001) 213-229.

[5] J. C. Cha, J. H. Cheon, An Identity-Based Signature from Gap Diffie-Hellman Groups. Proceedings of PKC, Lecture Notes in Computer Science, Vol. 2567, (2003) 18-30.

[6] Y. Choie and E. Lee, Implementation of Tate pairing on hyperelliptic curves of genus 2, Information security and cryptology—ICISC 2003, 97–111, Lecture Notes in Comput. Sci., 2971, Springer, Berlin, (2004).

[7] D. Cox, J. Little and D. O'Shea, Ideals, varieties, and algorithms : an introduction to computational algebraic geometry and commutative algebra : with 91 illustrations, New York : Springer, (1997).

[8] I. Duursma, Class numbers for hyperelliptic curves. In: Arithmetic, Geometry and Coding Theory. eds. Pellikaan, Perret, Vladuts, pp. 45-52, publ. deGruyter, Berlin, (1996).

[9] I. Duursma and H. Lee, Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$, Advances in cryptology-AsiaCrypt 2003, LNCS 2894, (2003) 111–123.

[10] G. Frey, H.-G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp. 62, no. 206, (1994) 865–874.

[11] S. D. Galbraith, K. Harrison, D. Soldera, Implementing the Tate pairing. Algorithmic Number Theory Symposium, ANTS-V, Lecture Notes in Computer Science, Vol. 2369, Springer-Verlag, (2002) 324–337.

[12] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for PolynomialComputations. Centre for Computer Algebra, University ofKaiserslautern (2005). `http://www.singular.uni-kl.de`.

[13] F. Hess, Exponent group signature schemes and efficient identity based signature schems based on pairing, Proceedings of the Workshop Selected Areas in Cryptology, SAC, Aug. (2002).

[14] A. Joux, A one round protocol for tripartite Diffie-Hellman. Proceedings of Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, Vol. 1838, Springer-Verlag, (2000) 385-394.

[15] A. Karatsuba and Y. Ofman, Multiplication of Multidigit Numbers on Automata. Sov. Phys.-Dokl. (Engl. transl.), **7**, No. 7, (1963) 595-596.

[16] Knuth, The Art of Computer Programming, Vol. II, Addison Wesley, (2004).

[17] N. Koblitz, Algebraic Aspects of Cryptography, Springer-Verlag, (1998).

[18] N. Koblitz and A. Menezes, Pairing-based cryptography at high security levels. Proceedings of the Tenth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science, 3796, (2005) 3-36.

[19] T. Lange, Formulae for arithmetic on genus 2 hyperelliptic curves. Appl. Algebra Engrg. Comm. Comput. 15, no. 5, (2005) 295-328.

[20] E. Lee and Y. Lee, Tate pairing computation on the divisors of hyperelliptic curves for cryptosystems. Cryptology eprint Atchives, Available at http://eprint.iacr.org, 2005, No. 2005/166

[21] D. Mumford, Tata Lectures on Theta II, Birkhauser, (1984).

[22] K.G. Paterson, ID-based signature from pairings on elliptic curves, Electronics Letters. Vol. 38 (18), (2002) 1025-1026.

[23] J. Pelzl, T. Wollinger and C. Paar, Low cost security: explicit formulae for genus-4 hyperelliptic curves. Selected areas in cryptography, Lecture Notes in Comput. Sci., 3006, Springer, Berlin, (2004) 1–16.

[24] V. Shoup, A library for doing number theory, Software, 2001; see http://www.shoup.net/ntl/.

[25] N. Smart, On the Performance of Hyperelliptic Cryptosystems, Advances in Cryptology: Proceedings of Eurocrypt'99, LNCS 1592, Springer-Verlag, (1999) 165-175.

[26] N.P. Smart, An identity based authentication key agreement protocol based on pairing, Electronics Letters, Vol 38, pp 630-632, (2002).

[27] C. K. Yap, Fundamental Problems in Algorithmic Algebra, Oxford University Press, (2000).

Appendix.

In Section 5, by searching for good candidates for ℓ and n from $n = 29$ through $n = 79$, we find the following:

When $n = 29$, for $H_{(-1)}$ curve,

$\ell = 29542758054398104450874217525165651042521871765435101109943075021065009 7$.

When $n = 43$, for $H_{(-1)}$ curve,

$\ell = 53718618569186388018821703986374275351705576366850017552481452390195758887874407533286287888356386446 7$.

When $n = 47$, for $H_{(-1)}$ curve,

$\ell = 13749772461004425111203179773331321128112017469863375270022695103034065149004498912831678964830$
$78087313972998213 3$.

When $n = 73$, for $H_{(+1)}$ curve,

$\ell = 10553398064514656199046818606065495176614662671222319372367416319801315889940362184197553323184 6$
$99007812855786020479789551940934976512907234753096204258803335765166769800421495325836 47$.

Table 5: h_D formula for $7D$

Input	$D = [U_D, V_D] \in J_{H_d}(k)$	Cost
Output	$\hat{h}_{\tilde{D}}(\hat{X}, Y) = \delta_1 Y^3 + s(\hat{X})Y^2 + t(\hat{X})Y + \delta_{16}(\hat{X})$ $s(\hat{X}) = \delta_2\hat{X}^4 + \delta_3\hat{X}^3 + \delta_4\hat{X}^2 + \delta_5\hat{X} + \delta_6$ $t(\hat{X}) = \delta_7\hat{X}^8 + \delta_8\hat{X}^7 + \delta_9\hat{X}^6 + \delta_{10}\hat{X}^5 + \delta_{11}\hat{X}^4 + \delta_{12}\hat{X}^3 + \delta_{13}\hat{X}^2 + \delta_{14}\hat{X} + \delta_{15}$	
δ_1	$(\tilde{v}_2\tilde{u}_0(\tilde{v}_2(\tilde{v}_2\tilde{u}_0) + 3\tilde{v}_1(\tilde{v}_0 + 2\tilde{v}_2\tilde{u}_1)) + \tilde{v}_1^2(\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0) + \tilde{v}_0(\tilde{v}_2\tilde{u}_1 - \tilde{v}_0)^2)^7$	$8m + 2s$
δ_2	$(4(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0)(-\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) - \tilde{v}_1(3\tilde{v}_2\tilde{u}_0 + \tilde{v}_1\tilde{u}_1))^7$	$3m$
δ_3	$(-2\tilde{v}_2\tilde{u}_0(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) + \tilde{v}_1(2\tilde{v}_2\tilde{u}_0 + \tilde{v}_0\tilde{u}_1))^7$	$2m$
δ_4	$(3\tilde{v}_2^2\tilde{u}_0^2 + \tilde{v}_1\tilde{u}_0(-2\tilde{v}_2\tilde{u}_1 + 3\tilde{v}_0) + \tilde{v}_0\tilde{u}_1(2\tilde{v}_2\tilde{u}_1 - 2\tilde{v}_0))^7$	$2m + 1s$
δ_5	$(\tilde{v}_2\tilde{v}_0(2\tilde{v}_1\tilde{v}_0 - 3\tilde{u}_1\tilde{u}_0) - \tilde{v}_1\tilde{u}_1(\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0) + 2\tilde{v}_0^2\tilde{u}_0)^7$	$3m$
δ_6	$(2\tilde{u}_0(\tilde{v}_2\tilde{u}_0)(\tilde{v}_2\tilde{u}_1 - 2\tilde{v}_0) - (\tilde{v}_0\tilde{u}_1 + \tilde{v}_1\tilde{u}_0)(4\tilde{v}_1\tilde{u}_0 + 2\tilde{u}_1(\tilde{v}_0 - \tilde{v}_2\tilde{u}_1)))^7$	$4m$
δ_7	$(-2\tilde{v}_2\tilde{u}_1 + 3\tilde{v}_0)^7$	$0m$
δ_8	$(2\tilde{v}_2\tilde{u}_0 - \tilde{v}_1\tilde{u}_1)^7$	$0m$
δ_9	$(2\tilde{u}_1(\tilde{v}_0 - \tilde{v}_2\tilde{u}_1) + 2(2\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0))^7$	$0m$
δ_{10}	$(\tilde{u}_1(2\tilde{v}_2\tilde{u}_0 + \tilde{v}_1\tilde{u}_1) + \tilde{v}_0\tilde{u}_0)^7$	$1m$
δ_{11}	$(\tilde{u}_1^2(-2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) + \tilde{u}_0(\tilde{v}_2\tilde{u}_0 + 2\tilde{v}_1\tilde{u}_1))^7$	$2m + 1s$
δ_{12}	$(\tilde{u}_0(3\tilde{u}_1(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) - \tilde{v}_1\tilde{u}_0))^7$	$1m$
δ_{13}	$(\tilde{u}_0^2(-\tilde{v}_2\tilde{u}_1 + 2\tilde{v}_0) + 3\tilde{u}_1^2(\tilde{v}_1\tilde{u}_0 - \tilde{v}_0\tilde{u}_1))^7$	$2m + 1s$
δ_{14}	$(2\tilde{u}_0^2(\tilde{v}_2\tilde{u}_0 + 2\tilde{v}_1\tilde{u}_1) + 3(\tilde{u}_0\tilde{v}_0)\tilde{u}_1^2)^7$	$2m$
δ_{15}	$(\tilde{u}_1(\tilde{u}_0^2(-\tilde{v}_2\tilde{u}_1 + 2\tilde{v}_0) + 3\tilde{u}_1^2(\tilde{v}_1\tilde{u}_0 - \tilde{v}_0\tilde{u}_1)) + \tilde{u}_0^2(2\tilde{u}_1(\tilde{v}_2\tilde{u}_1 - \tilde{v}_0) + 4(\tilde{v}_0\tilde{u}_1 + \tilde{v}_1\tilde{u}_0))^7$	$2m$
Total cost	Notation: m denotes a multiplication in $\mathbb{F}_{7^n}$, and s a squaring in $\mathbb{F}_{7^n}$.	$32m + 5s$

Table 6: $H_{D,E}$ formula complexity counting

i	ith coefficient of $H_{\tilde{D}}$	Cost
12	-1	0
10	$\delta_7 v_{E,2} + 3u_1^7$	$1m$
9	$\delta_7 v_{E,1} + \delta_8 v_{E,2} + 3u_0^7$	$2\tilde{m}$
8	$\delta_2 v_{E,2}^2 + \delta_7 v_{E,0} + \delta_8 v_{E,1} + \delta_9 v_{E,2} + u_1^{14}$	$1m + 1s + 2\tilde{m}$
7	$2\delta_2 v_{E,1} v_{E,2} + \delta_3 v_{E,2}^2 + \delta_8 v_{E,0} + \delta_9 v_{E,1} + \delta_{10} v_{E,2} + 2u_0^7 u_1^7$	$2m + 3\tilde{m}$
6	$\delta_1 v_{E,2}^3 + 2\delta_2 v_{E,0} v_{E,2} + \delta_2 v_{E,1}^2 + 2\delta_3 v_{E,1} v_{E,2} + \delta_4 v_{E,2}^2 + \delta_9 v_{E,0}$ $+\delta_{10} v_{E,1} + \delta_{11} v_{E,2} + u_0^{14} + 3u_1^{21}$	$3m + 4\tilde{m} + 1s$
5	$3\delta_1 v_{E,1} v_{E,2}^2 + 2\delta_2 v_{E,0} v_{E,1} + 2\delta_3 v_{E,0} v_{E,2} + \delta_3 v_{E,1}^2 + 2\delta_4 v_{E,1} v_{E,2}$ $+\delta_5 v_{E,2}^2 + \delta_{10} v_{E,0} + \delta_{11} v_{E,1} + \delta_{12} v_{E,2} + 2u_0^7 u_1^{14}$	$7\tilde{m} + 1m$
4	$3\delta_1 v_{E,0} v_{E,2}^2 + 3\delta_1 v_{E,1}^2 v_{E,2} + \delta_2 v_{E,0}^2 + 2\delta_3 v_{E,0} v_{E,1} + 2\delta_4 v_{E,0} v_{E,2} + \delta_4 v_{E,1}^2$ $+2\delta_5 v_{E,1} v_{E,2} + \delta_6 v_{E,2}^2 + \delta_{11} v_{E,0} + \delta_{12} v_{E,1} + \delta_{13} v_{E,2} + 2u_0^{14} u_1^7 - u_1^{28}$	$5\tilde{m} + 2m$
3	$-\delta_1 v_{E,0} v_{E,1} v_{E,2} + \delta_1 v_{E,1}^3 + \delta_3 v_{E,0}^2 + 2\delta_4 v_{E,0} v_{E,1} + 2\delta_5 v_{E,0} v_{E,2} + \delta_5 v_{E,1}^2$ $+2\delta_6 v_{E,1} v_{E,2} + \delta_{12} v_{E,0} + \delta_{13} v_{E,1} + \delta_{14} v_{E,2} + 3u_0^{21} + 3u_0^7 u_1^{21}$	$1m' + 5\tilde{m} + 1m$
2	$(3\delta_1 v_{E,0}^2 v_{E,2} + 3\delta_1 v_{E,0} v_{E,1}^2 + \delta_4 v_{E,0}^2 + 2\delta_5 v_{E,0} v_{E,1} + 2\delta_6 v_{E,0} v_{E,2} + \delta_6 v_{E,1}^2$ $+\delta_{13} v_{E,0} + \delta_{14} v_{E,1} + \delta_{15} v_{E,2} + u_0^{14} u_1^{14}$	$1m' + 2\tilde{m} + 1m + 1s$
1	$3\delta_1 v_{E,0}^2 v_{E,1} + \delta_5 v_{E,0}^2 + 2\delta_6 v_{E,0} v_{E,1} + \delta_{14} v_{E,0} + \delta_{15} v_{E,1} + 3u_0^{21} u_1^7$	$1m + 1m' + 1\tilde{m}$
0	$\delta_1 v_{E,0}^3 + \delta_6 v_{E,0}^2 + \delta_{15} v_{E,0} - u_0^{28}$	$1m' + 1s$
	Total cost	$4s + 13m + 31\tilde{m} + 4m'$